Department of Information Engineering and Computer Science

Master's Degree in

Information and Communications Engineering

FINAL DISSERTATION

Support Vector Regression with D-wave Quantum Annealer

Supervisor

Prof. Farid Melgani

Co-Supervisor

Dr. Gabriele Cavallaro

Student

Edoardo Pasetto

Academic year 2020/2021

Contents

Chapter 1

Abstract

The objective of this this master's thesis is to propose a method to optimize the cost function related to the training phase of a Support Vector Machine (SVM) for regression. The novelty in the optimization process that was developed in this study consists in the usage of Quantum Annealing (QA) for the minimization of the cost function. The hardware used for the annealing process is the D-wave Advantage system that was released in 2020. The implementation proposed in this work, that will be referred to as Quantum Support Vector Regression (QSVR), was tested on simple test cases of one variable functions in order to carry out a preliminary assessment on toy examples. The functions considered in the experimental analysis were a sine function, a sinc function and a triangular function. The number of training samples was kept low in each experiment beacuase the objective was to study the model in simple cases. Moreover, the hyperparameters were chosen in such a way to favor overfitting. In the optimization process the annealer provides different solutions and therefore 5 different methods to combine such solutions were proposed and then compared with the traditional SVR as a reference in the experiments. One interesting fact that emerged during the experimentation is that the different methods for combining the solutions performed very differently depending on the function considered for the testing. This suggests that there is not a best method overall but it should instead be chosen specifically based on the test case. Further research in the future will study the generalization capabilities of the QSVR and will also consider more complex functions and test cases for the training and the testing phase.

Chapter 2

Introduction

Over the last years the field of machine learning (ML) has become of great importance in many diverse applicative fields such as medicine [21], economics [31] and agriculture [27]. Broadly speaking the objective of ML is to build algorithms that are able to learn without being explicitly programmed to do so [24], [19]. In this thesis the focus is on regression problems, that consist in determining the relationship between one or more independent variables and one dependent variable. In particular, the objective of this work is the study of Support Vector Machines (SVM) applied to regression. SVMs are well-known supervised learning models that were introduced in the 1990s for the classification case [6] and for the regression case [9]. The algorithm of Support Vector Machines for both classification and regression requires the solving of a constrained optimization problem. In this work a method for solving such problem for the regression case with the usage of QA is proposed. In a similar way as was done in [29], for the sake of clarity, the original implementation of Support Vector Regression (SVR) will be referred as *classical* or *traditional* SVR, whereas the proposed QA-based implementation will be referred as *quantum* SVR or QSVR. Quantum Annealing is a metaheuristic, i.e. a "general algorithmic frameworks, often nature-inspired, designed to solve complex optimization" [3], used for solving multivaribale optimization problems by exploiting quantum fluctations as a search strategy. QA was first introduced in [2] and [10] and over the last years has established itself as an active field of research and a powerful asset for machine learning problems [14]. In QA the problem is encoded into the Hamiltonian of a quantum system and the physical model requires that such problem is a Quadratic Unconstrained Optimization (QUBO) problem. In a QUBO problem therefore the variables are binary and can take as value either 0 or 1 and can only be present in the equation as linear or quadratic terms. The QUBO problems must be unconstrained, however, it is possible to modify the target function in order to enforce the constraints in a implicit way. For example, in [29] this was done by adding penalty terms to the cost function whose influence is regulated by some hyperparameters. The quantum annealer returns different solutions, whose number is selected by the user, and in this work were proposed different methods to efficiently combine the such candidate solutions to produce the final solution. The hardware architecture that was used for the Quantum Annealing is the D-wave Advantage systems, which was released in 2020 and has 5000 qubits, more than double than its predecessor Dwave 2000Q. Moreover, the new Advantage system has a new hardware topology that gives a greater connectivity for each physical qubit. In [30] was shown that Advantage was able to outperform 2000Q in some computational tasks. Since QA requires that the problem is formulated as a QUBO, the main challenge when dealing with QA-based machine learning models is often to reformulate the optimization problem as a QUBO in an effective way [14]. In [12] QA was applied to maximum likelihood estimation problems, experimental design and matrix inversion. One of the first attempt to apply QA to machine learning was done in [20] in which the optimization of a classifier based on an ensemble of perceptrons is carried out with QA. It has also been recently applied in the context of Deep Learning: for instance, in [1] it was used for the optimization of a Deep Neural Network. Some implementations of classification algorithms with QA have have already proven to be able to outperform classical ones in some classification tasks related to computational biology [17]. In the context of SVM a method for solving the optimization problem for the classification case has already been proposed in [7] and [29]. Regression techniques based on the implementation of linear

regression with QA have already been proposed in [8] and [25], however, to the best of my knowledge, a regression technique based on the implementation of Support Vector Regression with QA has yet to be proposed.
Having at disposal an efficient and reliable QSVR algorithm would be of great importance for practical applications in different fields. The objective of this thesis is therefore to study a first implementation of a QA-based SVR and to lay out the basis for future research on this particular model. This thesis is structured as follows: in chapter 3 the theory and the mathematical formulation of Support Vector Machine is briefly described. In chapter 4 the a brief introduction on quantum computing, both gate-based and adiabatic is provided. In chapter 5 the reformulation of the original SVR optimization problem as a QUBO and the proposed methods for combining the solutions of the annealer are described and in chapter 6 the results of the experimental validation are illustrated and discussed. Finally, in chapter 7 the main conclusions as well as future prospect regarding the proposed implementation of QSVR are outlined and chapter 8 is dedicated to the acknowledgments.

Chapter 3

Support Vector Machines

3.1 Introduction

Support vector machines are one of the most popular supervised algorithms of machine learning mainly used for classification and regression problems. They were first introduced in 1992 in [6]. The reason for their popularity is to be found in the fact that the determination of the model parameters amounts to a convex optimization problem, therefore any local solution corresponds to a local one. Moreover, the mathematical structure of the problem allows the usage of the so-called "kernel trick", thus highly increasing the algorithm's expressive potential [4]. Another important feature of support vector machines is that, due to their formulation, they have sparse solutions, thus highly reducing the computational burden.
The functioning of support vector machines for both classification and regression purposes is now briefly illustrated.

3.2 SVM for classification

For the mathematical derivation as well as the notation used in this section I referred to [4] (pp. 326-330). In the classification the algorithm takes as input a dataset comprised of N datapoints $(\mathbf{x_n}, t_n)$, where $\mathbf{x}_n$ is a vector of dimension d and t_n is a binary label such that $t_n \in \{-1, 1\}$. Support vector machines use the function

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{3.1}$$

where $\mathbf{w}$ is a weight vector, $\phi(\mathbf{x})$ is non-linear feature-space transformation and b is called bias parameter. The label is then assigned by evaluating the sign of equation (3.1). To determine the optimal values for the parameters of the hyperplane defined in equation 3.1 the concept of margin is introduced: it is defined as the smallest distance between the hyperplane and any of the training samples. In the most common implementation of SVM called Maximum Margin Classifier (MMC) the parameters are chosen so that the margin is maximized. Geometrically, the distance from a generic point x and a hyperplane defined by the formula $\mathbf{w}^T\mathbf{x} + b$ is given by $|y(\mathbf{x})|/||\mathbf{w}||$, however, in this formulation only solutions for which all points are correctly classified are considered, i.e. solutions for which the condition $t_n y(\mathbf{x}_n) > 0$ holds for each n. The distance can then be rewritten as:

$$\frac{t_n y(\mathbf{x}_n)}{||\mathbf{w}||} = \frac{t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)}{||\mathbf{w}||} \tag{3.2}$$

By using its definition the formula of the margin is given by:

$$\arg\max_{\mathbf{w},b}\left\{\frac{1}{||\mathbf{w}||}\min_n[t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b]\right\} \tag{3.3}$$

(Here the term $||\mathbf{w}||$ has been taken outside the minimization since it does not depend on n).
The problem is then reformulated in a way such that is easier to solve. In fact, it can be easily shown that by rescaling both $\mathbf{w}$ and b by the same scalar factor the distance $t_n y(\mathbf{x}_n)/||\mathbf{w}||$ remains

unchanged. Therefore it is arbitrarily possible to choose to set the value $t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)=1$ for the closest point to the margin. Because of this all the other points will satisfy the constraint

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)\geq 1,\ n=1,....,N \tag{3.4}$$

Then it can be shown that the problem amounts to solving:

$$\arg\min_{\mathbf{w},b}\left\{\frac{1}{2}||\mathbf{w}||^2\right\} \tag{3.5}$$

with the constraints defined by (3.4). Here the term b does not appear in the function to be minimized but it acts implicitly on the value of w through the constraints. To optimize this function it is necessary to introduce the lagrange multipliers $\alpha_n \geq 0$, one for each constraint. The corresponding Lagrangian function is:

$$L(\mathbf{w},b,\underline{\alpha})=\frac{1}{2}||\mathbf{w}||^2-\sum_{n=1}^{N}\alpha_n\{t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)-1\} \tag{3.6}$$

(Here the notation $\underline{\alpha}=(\alpha_1,....,\alpha_N)$ has been used). By imposing the derivatives with respect to $\mathbf{w}$ and b to be equal to 0 the following conditions are obtained:

$$\mathbf{w}=\sum_{n=1}^{N}\alpha_n t_n\phi(\mathbf{x}_n) \tag{3.7}$$

$$0=\sum_{n=1}^{N}\alpha_n t_n \tag{3.8}$$

By substituting them into (3.6) it is possible to obtain the dual form of the function to be minimized:

$$\tilde{L}(\mathbf{a})=\sum_{n=1}^{N}\alpha_n-\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m t_n t_m k(\mathbf{x}_n,\mathbf{x}_m) \tag{3.9}$$

with the constraints:

$$\alpha_n\geq 0\quad n=1,...,N \tag{3.10}$$

$$\sum_{n=1}^{N}\alpha_n t_n=0 \tag{3.11}$$

Where the kernel function has been defined as: $k(\mathbf{x}_n,\mathbf{x}_m)=\phi(\mathbf{x}_n)\phi(\mathbf{x}_m)$. Many different kernels can be used, but the most popular in machine learning and especially SVM are the sigmoid, the polynomial and the radial basis function (RBF) kernel [4].
The discriminant function then becomes:

$$y(\mathbf{x})=\sum_{n=1}^{N}\alpha_n t_n k(\mathbf{x},\mathbf{x}_n)+b \tag{3.12}$$

and the associated decision function is:

$$\text{sign}(y(\mathbf{x})) \tag{3.13}$$

An important aspect of the solution can be deduced by observing the complementary Karush Kuhn Tuecker(KKT) conditions: in fact the following three properties hold:

$$\alpha_n\geq 0 \tag{3.14}$$

$$t_n y(\mathbf{x}_n)-1\geq 0 \tag{3.15}$$

$$\alpha_n\{t_n y(\mathbf{x}_n)-1\}=0 \tag{3.16}$$

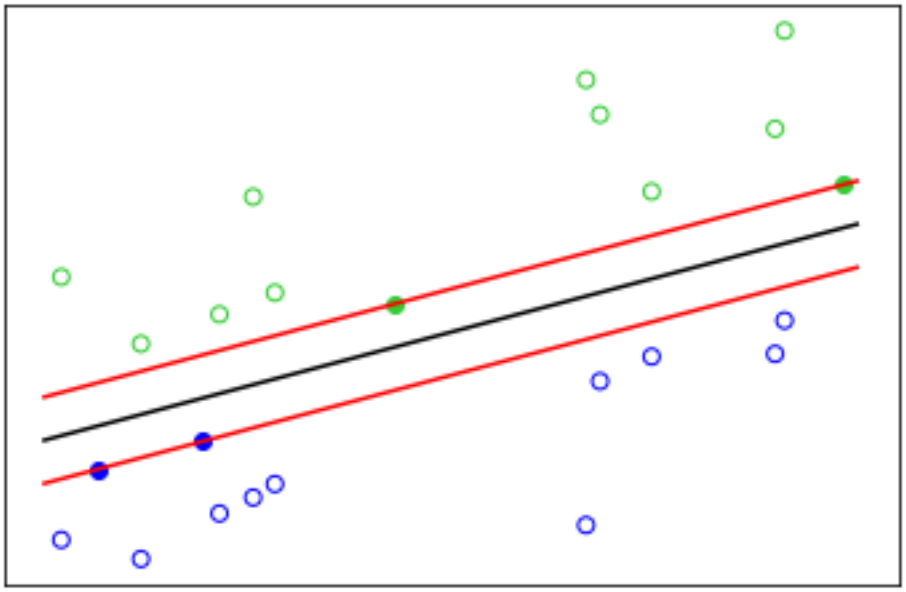


Figure 3.1: Graphical depiction of the hyperplane (black) in the feature space with the margins (red)

In particular, by examinating the last condition it can be seen that for any data point the associated multiplier α_n is either equal to 0 or $t_n y(\mathbf{x}_n) = 1$. This implies that only the points for which $t_n y(\mathbf{x}_n) = 1$, which are called support vectors, affect the prediction of new samples. This is of great importance for the applicability of SVM since it grants the sparsity of the solution.
Finally, to the determine the value of b it can be observed that, for any support vector $\mathbf{x}_n$, $t_n y(\mathbf{x}_n) = 1$. Therefore:

$$t_n \left(\sum_{m \in \mathcal{S}} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1 \tag{3.17}$$

where $\mathcal{S}$ denotes the set of the support vectors.
Even though only one support vector is needed, it is advisable to average over all of them in order to obtain a more numerically stable solution.

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \tag{3.18}$$

where $N_{\mathcal{S}}$ denotes the number of support vectors.

3.3 SVM for regression

After having introduced the Support vector machines for classification problems, their application to regression is now briefly described. The idea of using SVM for regression tasks was introduced in [9]. As was done for the previous section I referred to [4](pp.339-343) for the mathematical derivation of Support Vector Regression as well as the notation used. Generally, in a regression problem the task is to minimize a regularized function of the type:

$$\frac{1}{2} \sum_{n=1}^{N} \{y_n - t_n\}^2 + \frac{\lambda}{2} ||\mathbf{w}||^2 \tag{3.19}$$

Where here a least-square loss function has been considered but others can be used instead. However, in order to obtain sparse solution, a ϵ-insensitive error function is used: it produces a zero error if $|y(\mathbf{x}) - t| < \epsilon$, with ϵ being strictly positive. For the scope of this work the following cost function will be considered:

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t|, & \text{otherwise} \end{cases} \tag{3.20}$$

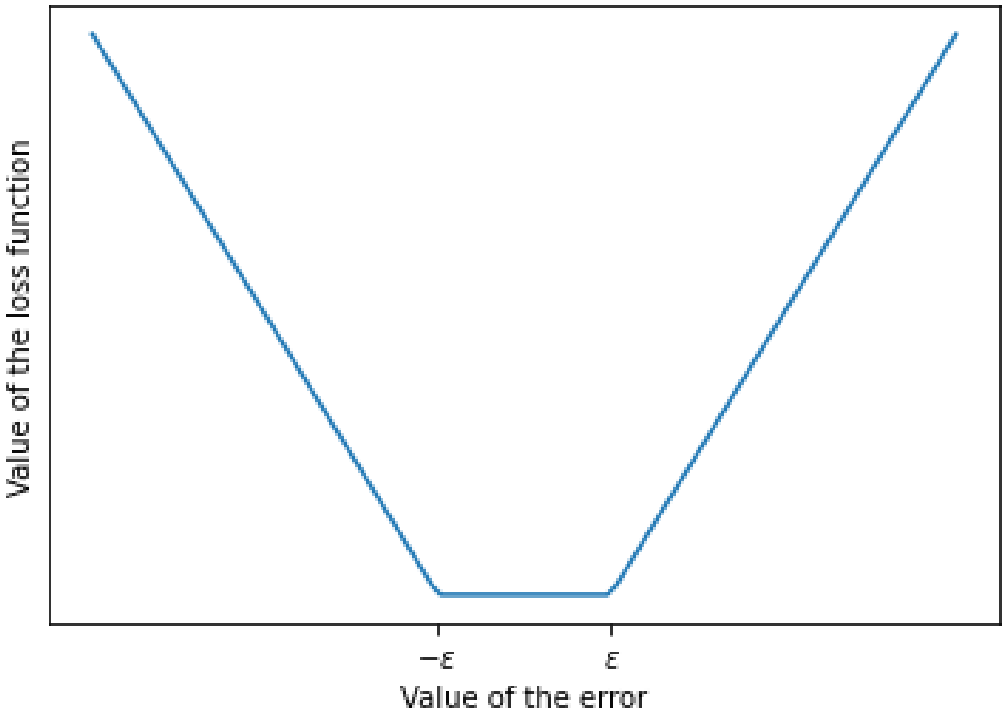


Figure 3.2: Graphical depiction of the considered error function

The error function therefore becomes:

$$C\sum_{n=1}^{N} E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2}||\mathbf{w}||^2 \tag{3.21}$$

Here C is used to indicate the regularization parameter. In this analysis $y(\mathbf{x})$ is given by (3.1).
To better solve the problem slack variables are introduced. In particular for each data point 2 slack variables, $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$ are defined. The situation where $\xi_n > 0$ arises when $t_n > y(\mathbf{x}_n) + \epsilon$, whereas $\hat{\xi}_n > 0$ corresponds to a situation where $t_n < y(\mathbf{x}_n) - \epsilon$. An example is illustrated in figure 3.3.
By introducing slack variables points can lie outside the ϵ-tube if such variables are different from zero. This allows us to rewrite the error function as:

$$C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}||\mathbf{w}||^2 \tag{3.22}$$

subject to the constraints:

$$\xi_n \geq 0 \tag{3.23}$$

$$\hat{\xi}_n \geq 0 \tag{3.24}$$

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n \tag{3.25}$$

$$t_n \geq y(\mathbf{x}_n) - \epsilon - \xi_n \tag{3.26}$$

For each $n \in \{1, \ldots, N\}$
As we did for the classification case the dual form of the cost function is obtained by setting the derivative with respect to $\mathbf{w}, b, \xi, \hat{\xi}_n$ to zero. It can be shown that the resulting function is:

$$\tilde{L}(\underline{\alpha}, \underline{\hat{\alpha}}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n - \hat{\alpha}_n)(\alpha_m - \hat{\alpha}_m)k(\mathbf{x}_n, \mathbf{x}_m) - \epsilon\sum_{n=1}^{N}(\alpha_n + \hat{\alpha}_n) + \sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n)t_n \tag{3.27}$$

subject to the constraints:

$$\sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n) = 0 \tag{3.28}$$

$$0 \leq \alpha_n \leq C \tag{3.29}$$

$$0 \leq \hat{\alpha}_n \leq C \tag{3.30}$$

Once the parameters are determined predictions of new input can be done using:

$$y(\mathbf{x}) = \sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n)k(\mathbf{x}, \mathbf{x}_n) + b \tag{3.31}$$

The value of b can be obtained from any point for which $0 < a_n < C$. Due to the constraints it must satisfy $\epsilon + y_n - t_n = 0$. Therefore the value of b is obtained through the formula:

$$b = t_n - \epsilon - \sum_{m=1}^{N}(\alpha_m - \hat{\alpha}_m)k(\mathbf{x}_n, \mathbf{x}_m) \tag{3.32}$$

As for the classification case it preferrable to average over different estimations of b to get a more stable solution:

$$b = \frac{1}{N_{\mathcal{S}}}\sum_{n\in\mathcal{S}}\left(t_n - \epsilon - \sum_{m=1}^{N}(\alpha_m - \hat{\alpha}_m)k(\mathbf{x}_n, \mathbf{x}_m)\right) \tag{3.33}$$

where,as was done for the classification case, $\mathcal{S}$ and $N_{\mathcal{S}}$ are used to denote the set of the support vectors and the number of them, respectively. An analogous procedure can also be done by considering points for which $0 < \hat{\alpha}_n < C$. Interesting insights can be observed by analyzing the KKT conditions:

$$\alpha_n(\epsilon + \xi_n + y_n - t_n) = 0 \tag{3.34}$$

$$\hat{\alpha}_n(\epsilon + \hat{\xi}_n - y_n + t_n) = 0 \tag{3.35}$$

$$(C - \alpha_n)\xi_n = 0 \tag{3.36}$$

$$(C - \hat{\alpha}_n)\hat{\xi}_n = 0 \tag{3.37}$$

By analyzing the first one it is possible to note that either $\alpha_n = 0$ or $\epsilon + \xi_n + y_n - t_n = 0$, implying that only the points that lie on the boundary of the tube ($\xi_n = 0$) or those who lie above it ($\xi_n > 0$) contribute to the prediction of new inputs. Analogous conclusions can be deduced from the second equation where $\hat{\alpha}_n > 0$ entails that the corresponding point either lies on the boundary or below it ($\hat{\xi}_n = 0$ and $\hat{\xi}_n > 0$ respectively). In the regression case the support vectors are those data points that lies in the boundary of the ϵ-tube or outside it. Finally, the constraints $\epsilon + \xi_n + y_n - t_n = 0$ and $\epsilon + \hat{\xi}_n - y_n + t_n = 0$ cannot be simultaneously satisfied because by considering the following:

$$\epsilon + \xi_n + y_n - t_n + \epsilon + \hat{\xi}_n - y_n + t_n = 0$$

$$2\epsilon = -(\xi_n + \hat{\xi}_n)$$

that is not possible since by definition $\epsilon > 0$ and $\xi_n \geq 0$, $\hat{\xi}_n \geq 0$.

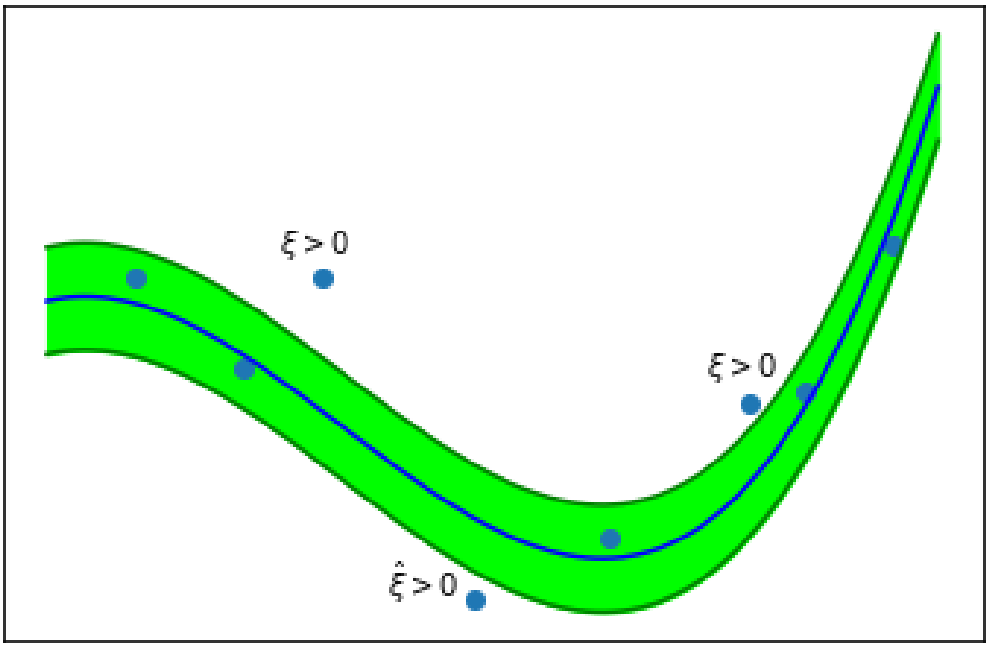


Figure 3.3: Graphical depiction of the target function along with the ϵ-insensitive tube and the training data points

Chapter 4

Quantum Computing

The first formal formulation of Quantum Computing was proposed in 1985 by David Deutsch [22]. In 1992 was proposed By Deutsch and Richard Josza the first quantum algorithm that was able to outperform the classical counterpart [23]. In the following years many other important quantum algorithms were proposed such as Shor's algorithm for factoring integers [28], Grover's search algorithm [13] and the Harrow-Hassidim-Lloyd algorithm for solving a linear system of equations [15]. Before discussing the principles of quantum computation the basics of quantum mechanics and the mathematical formalism necessary for the understanding will be briefly illustrated.

4.1 Basics of Quantum Mechanics

4.1.1 Postulates of Quantum Mechanics

Quantum mechanics, which was mainly developed in the 20^{th} century, is the physical theory used to describe physical systems at a microscopic scale. In fact, classical mechanics and classical electromagnetism fail to describe nature at a microscopic scale. This brief introduction to quantum mechanics will start by providing the postulates that describe the phenomenology of the theory. A deep analysis of quantum mechanics and its mathematical framework is beyond the scope of this thesis, in this section we will provide the reader with the sufficient knowledge to understand the basic principles of quantum mechanics in order to apply them to the understanding of quantum computing.

- A measurable physical quantity is defined as *observable* and we denote it with capital letters. The first postulate states that given an observable A, repeated measurements on it in the same physical condition (*state*) yield different results. The possible outcomes of the measurement process of the observable A are denoted with $\sigma(A) = \{a_1, \dots, a_n\} \in \mathbb{R}$, given the state ψ, the probability of obtaining a specific outcome follows the probability distribution $\mathbb{P}_{\psi}^{(A)}$ over $\sigma(A)$. For any observable A exists at least one state ψ_a such that: $\mathbb{P}_{\psi_a}^{(A)} = 1$.
- Given a state ψ and an observable A, if a measurement process over A is performed and the outcome is $a \in \mathbb{R}$, then the state of the system after the measurement process is ψ_a.
- There are some pairs of observables that cannot be simultaneosuly measured. Such observables are defined as incompatible.

4.1.2 Mathematical framework

The mathematical tools necessary for the understanding of quantum computing are now briefly illustrated. Generally, quantum mechanics is described with linear operators over infinite-dimension Hilbert spaces. However, for the scope of quantum computing only finite-dimension Hilbert spaces can be considered, which simplifies the mathematical formalism. The definition of Hilbert space is:

*A **Hilbert space** is defined by a pair consisting of a complex vector space H and a map $\langle \ | \ \rangle$, called inner product, defined over $H \times H \to \mathbb{C}$ satisfying the following properties*:

- is linear with respect tp the right entry
- is anti-linear with respect to the left entry
- $\langle\psi|\phi\rangle$=$\langle\phi|\psi\rangle^*$ $\quad \forall\ \psi, \phi \in H$
- $\langle\psi|\psi\rangle \geq 0 \quad \forall\psi \neq 0$

A basis of H $\{\psi_i\}_i$ is said to be *orthonormal* if it satisfies the condition: $\langle\psi_i|\psi_j\rangle = \delta_{ij}$, with δ_{ij} being the Kronecker delta. When considering a Hilbert space of finite-dimension n, after having fixed a basis of such space, it is possible to identify it with the complex vector space $\mathbb{C}^n$ and any linear operator on it can be identified with a $n \times n$ matrix. The set of linear operators over an Hilbert space H are indicated with the notation $\mathfrak{B}(H)$. For a linear operator A defined over the Hilbert space H we can define the *adjoint* operator of A, $A^\dagger$ such that :

$$\left\langle A^\dagger\psi \middle| \phi \right\rangle = \langle\psi|A\phi\rangle \qquad \forall\psi, \phi \in H \tag{4.1}$$

If for a given operator A the equality $A^\dagger = A$ the opertaor is called ***self-adjoint***. Given an operator A over H, the *trace* of A is defined as:

$$\mathrm{tr}(A) := \sum_i \langle\psi_i|A\psi_i\rangle \tag{4.2}$$
$$\text{with } \{\psi_i\}_i \text{ being an orthonormal basis of H}$$

For an operator A, its **spectrum** $\sigma(A)$ is defined as:

$$\sigma(A) := \{\lambda \in \mathbb{C} : A\psi = \lambda\psi, \psi \in H\backslash\{0\}\} \tag{4.3}$$

For a $\lambda \in \sigma(A)$, the *eigenspace* H_λ is defined as:

$$H_\lambda = \mathrm{span}\{\psi \in H : A\psi = \lambda\psi\} \tag{4.4}$$

With the definition of inner product $\langle \ | \ \rangle$ the *Dirac or bra-ket notation* was introduced. According to it, a unit vector of a Hilbert space H is denoted with the symbol *ket* $|\psi\rangle$, while a vector of the dual space is defined with the notation *bra* $\langle\psi|$. When considering vectors with their coordinate with respect to a given base it is possible to see that:

$$|\psi\rangle = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \qquad \langle\psi| = (c_1^*, \ldots, c_n*) \tag{4.5}$$

The values c_i are complex numbers and c_i^* are their complex conjugates. The inner product $\langle \ | \ \rangle$ of the Hilbert space is called *braket* and it can be seen as the product between a vector of the Hilbert space and another one from the corresponding dual space. Another important mathematical tools necessary for the understanding of quantum computing is the concept of tensor product:
*given the Hilbert spaces H_A and H_B and $\psi \in H_A$ and $S\phi \in H_B$, it is called **tensor product of vectors** the map $\psi \otimes \phi$ defined as:*

$$\begin{aligned} \psi \otimes \phi &: H_A \times H_B \to \mathbb{C} \\ \psi \otimes \phi(x; y) &:= \langle\psi|x\rangle_A \cdot \langle\phi|y\rangle_B \end{aligned} \tag{4.6}$$

The vector space defined by :

$$H_A \otimes H_B := \mathrm{span}\{\psi \otimes \phi,\ \psi \in H_A, \phi \in H_B\} \tag{4.7}$$

is called **tensor product of Hilbert spaces** and it is a Hilbert space itself with associated inner product defined as:

$$\begin{aligned} \langle \psi \otimes \phi | \psi' \otimes \phi' \rangle := \langle \psi | \psi' \rangle_A \cdot \langle \phi | \phi' \rangle_B \\ \psi, \psi' \in H_A, \phi, \phi' \in H_B \end{aligned} \tag{4.8}$$

If the Hilbert Space H_A has dimension n and H_B has dimension m, then $H_A \otimes H_B$ has dimension $n \cdot m$.

It is also possible to define the tensor product over linear operators: let $A \in \mathfrak{B}(H_A)$ and $B \in \mathfrak{B}(H_B)$, with $\mathfrak{B}(H)$ denoting the set of linear operators defined over H. The tensor product between A and B is defined as:

$$(A \otimes B)(\psi \otimes \phi) := A\psi \otimes B\phi \tag{4.9}$$

For the tensor product of linear operators holds the following property:

$$\begin{aligned} &\mathcal{B}(H_A) \otimes \mathcal{B}(H_B) := \mathcal{B}(H_A \otimes H_B) \\ &\text{with } \mathcal{B}(H_A) \otimes \mathcal{B}(H_B) = \text{span}\{A \otimes B : A \in \mathcal{B}(H_A), B \in \mathcal{B}(H_B)\} \end{aligned} \tag{4.10}$$

When considering two Hilbert spaces H_A and H_B of dimension m and n respectively and fixing a base for both of them it is possible to identify them with $\mathbb{C}^n$ and $\mathbb{C}^m$. Moreover, by fixing the bases the linear operators acting on them correspond to the matrices of dimension $n \times n$ and $m \times m$. The tensor product between an operator $A \in \mathfrak{B}(H_A)$ and an operator $B \in \mathfrak{B}(H_B)$ amounts to the Kronecker product between the corresponding matrices:

$$A \in \mathfrak{B}(H_A) = \begin{bmatrix} a_{11} & \cdot & \cdot & \cdot & a_{1n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix} \quad B \in \mathfrak{B}(H_B) = \begin{bmatrix} b_{11} & \cdot & \cdot & \cdot & b_{1m} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ b_{m1} & \cdot & \cdot & \cdot & b_{mm} \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdot & \cdot & \cdot & a_{1n}B \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1}B & \cdot & \cdot & \cdot & a_{nn}B \end{bmatrix} \tag{4.11}$$

It is now possible to provide a basic mathematical formulation of quantum mechanics that revolves around the following points:

- A quantum system is mathematically described a Hilbert space H
- Observables are defined by self-adjoint operators over H.
- pure states are equivalence classes of unit vectors in H such that:

$$\psi \sim \phi \Leftrightarrow \exists \alpha \in \mathbb{R} : \psi = e^{i\alpha}\phi \tag{4.12}$$

The last point introduces the notion of *pure state* that is a notion of state that encodes the maximum knowledge about a quantum system. In particular, every unit vector in a Hilbert space represents a pure state, however, the correspondence is not bijective since two different unit vectors represent the same state if the condition 4.12 holds. Therefore, it possible to identify the quantum states with bijective relation not the unit vectors $|\psi\rangle$ but the *1-dimensional orthogonal projectos* $|\psi\rangle\langle\psi| \in \mathfrak{B}(H)$. In fact, if two unitary vector differ for a multiplicative phase factor, and therefore are associated to the same state, the orthogonal projector will be the same:

$$\begin{aligned} \psi = e^{ia}\phi,\ \alpha \in \mathbb{R} \\ |\psi\rangle\langle\psi| = e^{i\alpha}e^{-i\alpha}|\phi\rangle\langle\phi| = |\phi\rangle\langle\phi| \end{aligned}$$

On the other hand, a *mixed state* ρ is defined as a convex combination of pure states:

$$\rho = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle \langle\psi_i| \tag{4.13}$$
$$\text{with } \lambda_i \geq 0 \; \forall i \text{ and } \sum_i \lambda_i = 1$$

With these notion it is now possible to analyse the postulates of quantum mechanics under a mathematical point of view:

- **Randomness of measurement outcomes**: each observable A is described by a linear operator and the possible outcomes of the measurement process are the elements of the spectrum of A $\sigma(A)$. If the system is in the pure state ψ the probabilty to obtain $a \in \sigma(A)$ as a measurement outcomes $\mathbb{P}_\psi(a)$ is given by:
$$\mathbb{P}_\psi(a) = \langle\psi|P_a\psi\rangle = \|P_a\psi\|^2 \tag{4.14}$$
$\{P_a\}_{a\in\sigma(A)}$ is called spectral measure and is defined as:
$$P_a = \sum_{i=1}^{\dim H_a} |\psi_i\rangle \langle\psi_i| \tag{4.15}$$
where H_a is the eigenspace of a and $\{\psi_i\}_{i=1,\ldots,\dim H_a}$ is an orthonormal basis of it. Therefore, the expectation value for the outcome of A while the system is in the state ψ, $\langle A\rangle_\psi$ is equal to:
$$\langle A\rangle_\psi = \sum_{a\in\sigma(A)} a\, \mathbb{P}_\psi(a) = \langle\psi|A\psi\rangle \tag{4.16}$$
If the system is in a mixed state ρ the probability to obtain a as a result and the expectation become:
$$\mathbb{P}_\rho(a) = \text{tr}(P_a\rho) \tag{4.17}$$
$$\langle A\rangle_\rho = \text{tr}(A\rho) \tag{4.18}$$

- **Post-measurement state**: If a measurement process of A is carried out on a system in the pure state ψ with outcome $a \in \sigma(A)$, then the state of system after such measurement becomes:
$$\psi_a = \frac{P_a\psi}{\|P_a\psi\|} \tag{4.19}$$
If the system is in a mixed state ρ, the state after the measurement becomes:
$$\rho_a = \frac{P_a\rho P_a}{\text{tr}(P_a\rho)} \tag{4.20}$$

- **Compatible and incompatible observables**: Two observables are said to be compatible when it is possible to commute them:
$$[A, B] := AB - BA = 0 \tag{4.21}$$
If the condititon 4.21 holds then we have that: $P_a^A P_b^B = P_b^B P_a^A \; \forall a \in \sigma(A)$ and $\forall b \in \sigma(B)$. Therefore, the joint probability $\mathbb{P}_\psi(A = a \; \wedge \; B = b)$ of measuring a for A and b for B.

The last mathematical definition that is needed for the introduction of quantum computing is that of composite system, which revolves around the concept of tensor product that was introduced before. Specifically, given two quantum systems S_A and S_B that are mathematically described by the Hilbert

spaces H_A and H_B, respectively, the composite quantum system H is described by the tensor product between H_A and H_B:

$$H = H_A \otimes H_B$$

A pure state $\Psi \in H_A \otimes H_B$ of the composite system that can be written as:

$$\Psi = \psi \otimes \phi \qquad \psi \in H_A,\ \phi \in H_B \tag{4.22}$$

is called **separable**, otherwise it is called **entangled**. The concept of entangled and separable states is of utmost importance for many algorithms in quantum computing. The characterization of separable and entangled states provided above only applies to pure states, now we will provide a generalization that also applies to mixed states. A quantum state ρ of the composite system $H_A \otimes H_B$ is separable if ρ can be expressed as:

$$\begin{aligned} &\rho = \sum_i \lambda_i \rho_i^{(A)} \otimes \rho_i^{(A)} \\ &\text{with } \lambda_i \geq 0\ \forall i,\ \sum_i \lambda_i = 1,\ \rho_i^{(A)} \in \mathcal{S}(H_A),\ \rho_i^{(B)} \in \mathcal{S}(H_B) \end{aligned} \tag{4.23}$$

Where the notation $\mathcal{S}(H)$ denotes the set of all states, both pure and mixed, of the Hilbert space H.

4.2 Quantum Circuit Model

4.2.1 Qubits and qubits registers

In this section will be provided an introduction to the quantum computing gate model. One of the first important notion of such model is the **qubit**. The qubit can be thought as a generalization of the concept of bit that is used in classical computers. A bit is, in fact, a basic unit of information corresponding to a two-sates system. The possible states are usually denoted with 0 and 1. In the mathematical framework outlined in the previous section the states of a system are represented with elements of a Hilbert space, therefore the states of the classical bit are formulated as basis vectors of a two-dimensional Hilbert space, $|0\rangle$ and $|1\rangle$, where:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.24}$$

A qubit is therefore a pure sate of the two-dimensional Hilbert Space and can be therefore expressed as a complex combination of the classical states $|0\rangle$ and $|1\rangle$. The general for of a qubit ψ is:

$$\psi = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{with } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1 \tag{4.25}$$

Quantum computers operate on quantum systems composed by multiple qubits called **qubit registers**. Such composite systems are mathematically described by using the notion of tensor product: specifically, a n-qubit register is defined by considering the tensor product of the n different Hilbert spaces and is therefore equivalent to the space $(\mathbb{C}^2)^{\otimes n}$. For example let us consider the 2-qubit register: the 2^2 classical states are given by the tensor prosuct of the single qubit states: $|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |0\rangle$ (often a compact notation of the form: $|\psi\phi\rangle$ is used to denote $|\psi\rangle \otimes |\psi\rangle$). The four possible states are:

$$|00\rangle = \begin{matrix} 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{matrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad |01\rangle = \begin{matrix} 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{matrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$|10\rangle = \begin{matrix} 0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{matrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad |11\rangle = \begin{matrix} 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{matrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

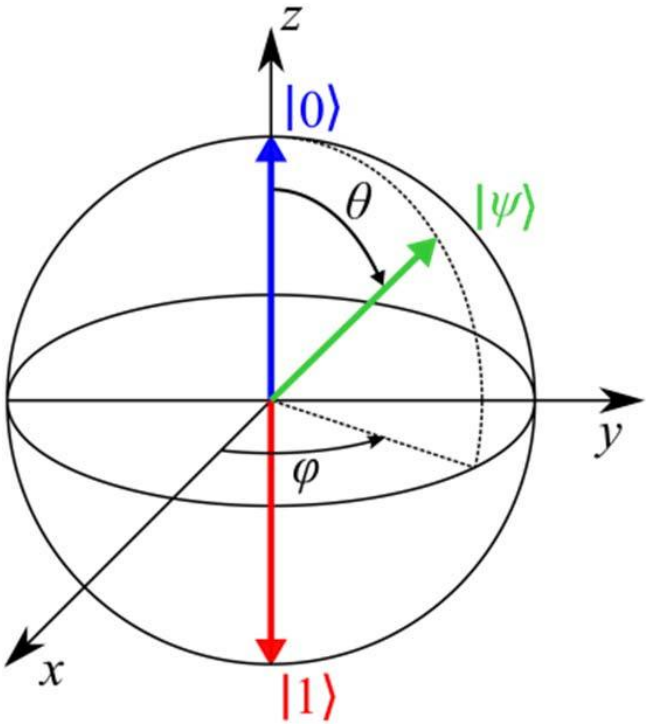


Figure 4.1: The Bloch Sphere, named after the physicist Felix Bloch, provides a graphical representation of the pure states of a qubit by encoding the parameters describing the qubit into the values of the angles θ and ϕ that are used to identify a unique location on the sphere. A point on the sphere at coordinates $\hat{\theta}\ \hat{\phi}$ corresponds to the qubit $\cos\frac{\hat{\theta}}{2}|0\rangle + e^{i\hat{\phi}}\sin\frac{\hat{\theta}}{2}|1\rangle$. Figure originally from [26]

A state of such quantum system is mathematically represented by a combination $\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle$ with $\alpha_i \in \mathbb{C}$ and $\sum_i |\alpha_i|^2 = 1$. The 2-qubit register it also useful to provide an example of the entaglement. Let us consider as example the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$: if a measurement process on the first qubit is performed and the result obtained is $|1\rangle$, then then it is possible to know already that the state of the second qubit is $|1\rangle$ without performing any measurement whatsoever. An analogous reasoning also applies if the result of the measurement on the first qubit yields $|0\rangle$ as a result.

4.2.2 Quantum gates and quantum circuits

As was done in the previous section the notion of quantum gates is introduced starting from the parallelism with classical ones. Logic gates represent the operation that are done on the bits to perform calculations. Let us consider, for instance, the case in which n input bits are processed by a logical gate that in turn gives as result m output bits. In the previous subsection we have indicated the states of the single bit as $|0\rangle$ and $|1\rangle$ and the states of a composite of bits as the tensor product of the bits involved. Therefore the n-bit input register and the m-bit output register are defined as column vectors of dimension 2^n and 2^m respectively. The logical gate is in turn mathematically described as a matrix of dimension $2^m \times 2^n$ that multiplies the $2^n \times 1$ input register resulting in a $2^m \times 1$ output. For example the well-known NOT gate, that has the formula: $\mathrm{NOT}|0\rangle = |1\rangle$, $\mathrm{NOT}|1\rangle = |0\rangle$ takes the form:

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

In the same way the AND gate that takes as input 2 bits and gives as result one bit takes the form:

$$AND = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In quantum computing the gates operating on qubits, that most of the time are in a superposition of states, must be **reversible**. For example, the AND gate is not reversible becsause it is not possible to determine the input of the function given the corresponding output, whereas the NOT gate belongs to the set of reversible operations. To ensure that a gate U is reversible the associated matrix must be **unitary**, i.e. $UU^\dagger = U^\dagger U = I$, where I is the identity matrix. Some of the most important quantum gates will be now described.

- **Hadamard gate**: This gate, named after the mathematician Jacques Hadamard, is one of the most important and most used in quantum computing.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.26}$$

  The application of the gate to the states $|0\rangle$ and $|1\rangle$ produces the states:

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{4.27}$$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{4.28}$$

  These sates are one of the most used in quantum algorithms.

- **Controlled NOT gate**: The gate, also denoted as CNOT for short, acts on 2 qubits and produces two qubits as output. The first input qubit is called control qubit and affects the operation that is carried out on the other one. If the control qubit is in the state $|0\rangle$ the identy operation is applied on the second qubit, whereas if it is in the state $|1\rangle$ the negation is performed. From the definition it easy to observe that the CNOT is its own inverse: it maps the state $|x, y\rangle$ to $|x, x \oplus y\rangle$, where $\oplus$ is the XOR operator. A second application of the CNOT to the state $|x, x \oplus y\rangle$ gives as output $|x, x \oplus y \oplus x\rangle = |x, y\rangle$. The corresponding unitary matrix is:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.29}$$

- **Phase gate**: The gate is parametrized by the parameter ϕ and acts on a single qubit according to the formula: $\alpha |0\rangle + \beta |1\rangle \rightarrow \alpha |0\rangle + e^{j\phi}\beta |1\rangle$. The associated unitary matrix is:

$$P_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{j\phi} \end{bmatrix} \tag{4.30}$$

The set consisting of the quantum gates H, CNOT and P_ϕ is called **universal** because any quantum circuit can be built as a composition of such gates.
Another important set of quantum gates is the set that consists of the Pauli matrices $(\sigma_x, \sigma_y, \sigma_z)$, which are named after the physicist Wolfgang Pauli, that take the form:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.31}$$

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{4.32}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{4.33}$$

As can be seen by the equations above, the σ_x is equivalent to the NOT gate and the σ_z is equivalent to the phase gate with $\phi = \pi$. The Pauli matrices will be used in section 4.4 to construct the problem Hamiltonian for the Quantum Annealing Process.

4.3 Adiabatic Quantum Computing

Adiabatic Quantum Computing is a paradigm of Quantum computation that was first proposed in 2000 by Fahri *et al.* [11]. In such a setting a quantum physical system is prepared in its lowest energy configuration state, that is also known as *ground state*, the other states are instead called *excited states*. The system's Hamiltonian is then gradually changed in such a way that the ground state of the final Hamiltonian corresponds to the solution of the considered optimization problem. If the mutation of the Hamiltonian occurs slowly enough the system remains in the ground state during the process according to the quantum adiabatic theorem formulated in 1928 by Born and Fock [5], that sates:

"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalues and the rest of the Hamiltonian spectrum".

The general idea of adiabatic quantum computing could be summarized as follows:

1. A quantum is in the ground system of a known Hamiltonian. Such Hamiltonian should be chosen in such a way that is easy to place the system in its ground state.
2. The Hamiltonian is gradually changed at a rate that is slow enough to guarantee that the adiabatic condition is met. The final Hamiltonian should "encode" the optimization problem, i.e. the ground state of the final Hamiltonian should correspond to the solution of the optimization problem.
3. When the mutation of the Hamiltonian is complete a measurement process of the quantum system is performed.

The adiabatic process is carried out for a time T. Afterwards the conditions on T such that the adiabatic theorem is satisfied will be provided. The Hamiltonian is time-dependent $H(t)$ and the evolution of a system in state ψ_0 is determined by the Schrödinger's equation:

$$\begin{aligned} &\{H(t)\}_{t\in[0,T]} \subset \mathfrak{B}(H) \\ &i\hbar\frac{d}{dt}\psi(t) = H(t)\psi(t) \\ &\text{with } \psi(0) = \psi_0 \text{ as initial condition} \end{aligned} \tag{4.34}$$

For convenience the Hamiltonian is usually reparametrized in the form: $\tilde{H}(s) = H(Ts)$ with $s \in [0, 1]$. The eigenstate problem therefore becomes:

$$\begin{aligned} &\tilde{H}(s)\,|I, s\rangle = E_I(s)\,|I, s\rangle \\ &\text{for } s \in [0, 1] \end{aligned} \tag{4.35}$$

Where the ground state of $\tilde{H}(s)$ was denoted with $|0, s\rangle$ and $E_0(s) = \min\sigma(\tilde{H}(s))$. By analyzing the adiabatic theorem under a mathematical point of view it is possible to see that:

$$\begin{aligned} \text{let } \lambda(s) = E_1(s) - E_0(s) \text{ for } s \in\ [0, 1] \text{ then} \\ \lim_{T\to+\infty} |\,\langle 0, 1|\psi(T)\rangle\,| = 1 \end{aligned} \tag{4.36}$$

with $\psi(T)$ being the solution of 4.34 at time $t = T$ considering as initial conditions $\psi(0) = |I = 0, s = 0\rangle$. Here we have considered $T \to \infty$, but if a determined value of T is considered instead, and therefore

a determined value of the speed at which the Hamiltonian is changed, it is possible to ensure that the system is in the ground state at the end of the process with a precision that follows the rule:

$$\begin{aligned} &\text{Given a T such that} \\ &T \geq \frac{1}{\epsilon} \cdot \frac{\max_s \left\| \frac{d}{ds} \tilde{H}(s) \right\|_{op}}{[\min_s \lambda(s)]^2} \qquad (4.37) \\ &\text{if } \lambda(s) = E_1(s) - E_0(s) > 0 \text{ for every } s \in [0,1] \text{ then:} \\ &\|P\psi(T)\|^2 \geq 1 - \epsilon^2 \qquad (4.38) \end{aligned}$$

Where $\epsilon \in (0,1)$ and the operator $||A||_{op}$, that is called operator norm is defined as: $|A||_{op} = \sup_{||\psi||=1} ||A\psi||$. The term $\psi(T)$ in equation 4.38 is the solution of the Schrödinger equation 4.34 at time $t = T$ considering as initial condition $\psi(0) = |0,0\rangle$. The term P, instead, denotes the orthogonal projector on the eigenspace $\mathcal{H}_{E_0(1)}$. The quantity $||P\psi(T)||^2$ is equal to the probability to find the system in the ground state at the end of the evolution process.

The procedure to apply adiabatic quantum computation to solve optimization problems will now be outlined. For the purposes of this work will be considered the optimization of a function f that takes as input n boolean variables. The system is thus described by a Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. The time-dependent Hamiltonian is then constructed considering a combination, that varies through time, of two time-independent Hamiltonians H_I and H_F:

$$H(t) = \left(1 - \frac{t}{T}\right) H_I + \frac{t}{T} H_F \qquad (4.39)$$

- H_I is the initial Hamiltonian with a well-known ground state
- H_F is the Hamiltonian encoding the problem

The time T is chosen in such a way that the adiabatic theorem is satisfied.

In the following subsection will be shown an example on how to apply AQC to the optimization of a boolean function.

4.3.1 Optimization of a 3-SAT problem with Adiabatic Quantum Computing

The general formulation of a 3-SAT problem over n boolean variables is:

$$B(z) = \bigwedge_{i=1}^{m} C_i(z_{i1}, z_{i2}, z_{i3}) \quad \text{with } z \in \{0,1\}^n \qquad (4.40)$$

In the formula 4.40 C_i denotes a clause that involves at most 3 of the n binary variables. An assignment of the n boolean variables $z^* \in \{0,1\}^n$ is a solution if all the m clauses are satisfied. To optimize the problem with AQC we first define, for each clause C_i an energy function h_i that is defined as:

$$h_i(z_{i1}, z_{i2}, z_{i3}) = \begin{cases} 0 \text{ if } (z_{i1}, z_{i2}, z_{i3}) \text{ satisfies } C_i, \\ 1 \text{ otherwise} \end{cases} \qquad (4.41)$$

From the functions $h_i, i \in \{1, \ldots, m\}$ it is possible to define the general energy function $h : \{0,1\}^n \to \{0, 1, \ldots, m\}$:

$$h(z) = \sum_{i=1}^{m} h_i(z_{i1}, z_{i2}, z_{i3}) \qquad (4.42)$$

Such function has the following properties:

1. $h \geq 0 \; \forall z$
2. $h = 0 \iff$ all clauses are satisfied

To transfer the problem to a quantum computing setting the boolean variables are encoded into the qubits using the basis encoding: $z_i \to |z_i\rangle$. For each clause C_i of the 3-SAT problem a operator H_i acting on the n-qubit Hilbert space is defined as:

$$H_i |z_1, \ldots, z_n\rangle := h_i(z_{i1}, z_{i2}, z_{i3}) |z_1, \ldots, z_n\rangle \tag{4.43}$$

The problem Hamiltonian thus becomes:

$$H_F = \sum_{i=1}^{m} H_i \tag{4.44}$$

In particular: $H_F |\psi\rangle = 0 \iff \psi$ is a superposition of states $|z_1, \ldots, z_n\rangle$ such that the corresponding binary variables assignment $(z_1, \ldots, z_n)$ are solutions of the problem, *i.e* satisfy all clauses.

4.4 Quantum Annealing

Quantum Annealing (QA) is a metaheuristic used to solve optimization problems. Like Adiabatic Quantum Computing it is based on the concept of time-dependent Hamiltonias but it differs from it for some aspects. As was done for AQC, the time-dependent Hamiltonian is expressed as a combination of two time-independent Hamiltonians:

$$H(t) = H_P + \gamma(t) H_D \tag{4.45}$$

The term H_P is the problem Hamiltonian, whose ground state represents the solution of the problem, while the term H_D is called kinetic term or disorder Hamiltonian. The function $\gamma(t)$ is a monotonically decreasing function to 0 that is defined over $[0, T]$. Therefore, like in AQC, there is again an evolution process, called *annealing*, of an Hamiltonian, but there are some important differences between the two models of computation:

1. Quantum Annealing, in contrast with AQC, is not a universal model of computation. QA is a quantum-based technique to solve optimization problems
2. In QA the system is not isolated
3. Unlike AQC, QA does not assume that the whole computation takes place in the ground state.

4.4.1 D-wave quantum annealer

For the implementation of the proposed model of QSVR the D-wave quantum annealer Advantage has been employed. In this section a brief descpription of the D-wave QA, especially focusing on the differences between its predecessor 2000Q, is provided. The information provided in this section for the description of the D-Wave Advantage system were taken from the official techinical report [18]. The main features of the Advantage system are the new topology for the problem embedding and the higher value of qubits and coupler, that are more than 5000 and 35000, respectively. A higher number of qubits directly translates to the possibility to solve more complex problems. Moreover, in the new design the number of couplers per quibits has increased from 6 to 15. A higher number of couplers for each qubit increases the *compactness* of the system, which relates to the length of the chain length. In fact, it was found that the chain lengths on the Advantage System were on average half as long as those of 2000Q. Having shorter chains is of great importance for the solution of the optimization problems because it has been observed that having chains of shorter length allows the

system to provide solutions of better quality given the same computational resources. In the D-Wave systems the equation 4.45 takes the form:

$$H(s) = -\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right) + \frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right) \tag{4.46}$$

In the equation 4.46 the first term is the initial Hamiltonian, whereas the second one represent the problem Hamiltonian. The functions A(s) and B(s) control the influence of each of the two Hamiltonians during the annealing process. They are controlled by the parameter s that is the normalized annealing time variable and whose values lie in the range $[0, 1]$. The terms $\hat{\sigma}_x^{(i)}$ and $\hat{\sigma}_x^{(i)}$ are the Pauli x and z matrices operating on the i-th qubit. The terms h_i and $J_{i,j}$ are called qubit biases and coupling strengths, respectively. Mathematically, the biases represent the values of the linear coefficients in the considered QUBO problem, whereas the coupling strength represent the quadratic terms.

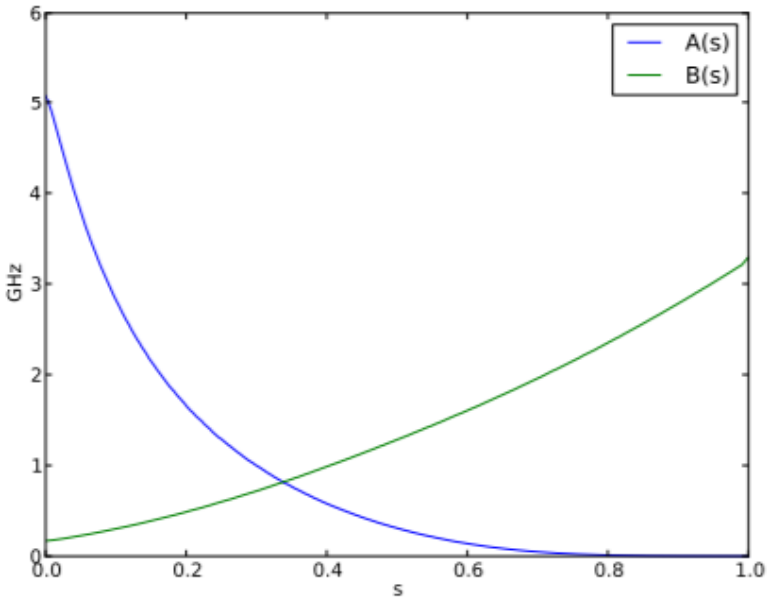


Figure 4.2: Graphical illustration of the functions A(s) and B(s) employed by the D-wave hardware for the annealing process. Figure originally from [16]

Chapter 5

Quantum Support Vector Regression

5.1 Optimization problems with the D-wave quantum annealer

To solve a optimization problem with a quantum annealer it is necessary to reformulate it as QUBO (Quadratic Unconstrained Binary Optimization) problem. The general formulation of such a problem is:

$$f(x) = \sum_{i \leq j} a_i Q_{i,j} a_j \tag{5.1}$$

Here, the terms a_i are the variables of the problem that can take as value either 0 or 1 and Q is a upper-triangular real-valued matrix. The reason why Q is upper triangular is to be found in the problem formulation: the element $Q_{i,j}$ of the matrix Q corresponds to the value of the coupler relative to the variables a_i and a_j if $i \neq j$ or to coefficient of the linear term of the variable a_i if $i = j$. The elements $Q_{i,j}$ and $Q_{j,i}$ refer to the same mathematical quantity and thus only one of them is needed to completely describe the problem. Moreover, the reason why the quadratic terms a_i^2 are not considered is due to the fact that the binary variables can only take as value either 0 or 1 and therefore the following equality holds: $a_i^2 = a_i$.
In section 4.4.1 it is described how the coefficients of the QUBO problem are related to the system Hamiltonian. Even though the QUBO problem is intrinsically unconstrained it is possible to run constrained optimization problems by modifying the cost function in such a way that the solution autmatically enforce the constraints. For instance, as will be discussed more in detail in the following section, in our case this is done by adding some penalty terms to the cost function whose strength is regulated by a hyperparameter in order to enforce the constraints.

5.2 Turn the optimization problem into a QUBO problem

It will now be shown how to turn the optimization problem of the Support vector regression into a instance of a QUBO problem in order to solve it with the quantum annealer. The procedure will start by analyzing the Support vector regression cost function minimization problem 3.27 subject to the constraints 3.28, 3.29 and 3.30 that are rewritten here for convenience.

$$\tilde{L}(\underline{\alpha}, \underline{\hat{\alpha}}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} (\alpha_n - \hat{\alpha}_n)(\alpha_m - \hat{\alpha}_m) k(\mathbf{x}_n, \mathbf{x}_m) + \epsilon \sum_{n=1}^{N} (\alpha_n + \hat{\alpha}_n) - \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) t_n$$

$$\sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) = 0$$

$$0 \leq \alpha_n \leq C$$

$$0 \leq \hat{\alpha}_n \leq C$$

To express the problem as a QUBO one the encodings introduced in [7] and [29] for the coefficients $\underline{\alpha}$ and $\underline{\tilde{\alpha}}$ are applied. In the regression case, however, for each training sample $\mathbf{x}_n$ 2 Lagrange multipliers are needed. The number of the logical qubits of the problem will be $2KN$, where N is the number samples and K is the number of qubits used to encode each coefficient α_n or $\hat{\alpha}_n$. In the proposed implementation the assignment of the variables is arranged as follows: the first KN qubits are used to encode the N α_n multipliers, whereas the remaining KN are used to encode the N $\hat{\alpha}_n$ coefficients. The encoding formulae therefore are given by:

$$\alpha_n = \sum_{i=0}^{K} B^i a_{Kn+i} \tag{5.2}$$

$$\hat{\alpha}_n = \sum_{i=0}^{K} B^i a_{K(N+n)+i} \tag{5.3}$$

where B is the basis used for the encoding. If the usage of negative exponent for the basis is needed the formulae 5.2 and 5.3 are reformulated by introducing a parameter k_0:

$$\alpha_n = \sum_{i=0}^{K-1} B^{i-k_0} a_{Kn+i} \tag{5.4}$$

$$\hat{\alpha}_n = \sum_{i=0}^{K-1} B^{i-k_0} a_{K(N+n)+i} \tag{5.5}$$

In this section, for simplicity, the mathematical model will be developed considering the encoding equations functions 5.2 and 5.3, however, the procedure considering the encoding equations 5.4 and 5.5 is the same and the final results only differ for the exponent of the base B in the equations. The constraints to be satisfied are: 3.28, 3.29, and 3.30. The last 2 of them are also known as box constraints. Since $a_n \in \{0, 1\}$ and $\hat{a}_n \in \{0, 1\}$ for $n \in \{1, \ldots, N\}$ the lower bound for the variables α_n and $\hat{\alpha}_n$ is always satisfied for such constraints. Moreover, by choosing:

$$C = \sum_{i=0}^{K-1} B^i \tag{5.6}$$

the upper bound is always upheld as well. To enforce the first constraint 3.28 a square-penalty term that is controlled by the hyperparameter ξ is added to the lagrangian function:

$$\xi \Big(\sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) \Big)^2 \tag{5.7}$$

The problem formulation also requires that for each sample either α_n or $\hat{\alpha}_n$ is equal to 0. To enforce such constraint a linear penalty term, whose strength is controlled by the hyperparameter β is introduced:

$$\beta \Big(\sum_{n=1}^{N} \alpha_n \hat{\alpha}_n \Big) \tag{5.8}$$

The equation 5.8 exploits the fact that since at least one of the coefficients α_n or $\hat{\alpha}_n$ must be 0, their product must be equal to 0 as well. The constraint is applied to the whole sum of the products $\alpha_n \hat{\alpha}_n$, however, since each of the α_n and $\hat{\alpha}_n$ is non-negative according to the chosen encoding equations, placing a constraint on their sum of each product to be equal to 0 enforce each single product to be equal to 0.

Therefore, by adding 5.9d and 5.8 to 3.27 and considering the encoding 5.2 and 5.3 the final cost function to minimize is obtained. To keep the notation simple each term of the cost function will

be considered individually and then the final result will be obtained by adding them together. The equations then are:

$$\frac{1}{2}\sum_n\sum_m\left(\sum_i B^i a_{Kn+i}-\sum_j B^j a_{K(N+n)+j}\right)\left(\sum_i B^i a_{Km+i}-\sum_j B^j a_{K(N+m)+j}\right)k(\mathbf{x}_n,\mathbf{x}_m) \tag{5.9a}$$

$$\epsilon\sum_n\left(\sum_i B^i a_{Kn+i}+\sum_j B^j a_{K(N+n)+j}\right) \tag{5.9b}$$

$$-\sum_n\left(\sum_i B^i a_{Kn+i}-\sum_j B^j a_{K(N+n)+j}\right)t_n \tag{5.9c}$$

$$\xi\left(\sum_n\left(\sum_i B^i a_{Kn+i}-\sum_j B^j a_{K(N+n)+j}\right)\right)^2 \tag{5.9d}$$

$$\beta\left(\sum_n\left(\sum_i B^i a_{Kn+i}\right)\left(\sum_j B^j a_{K(N+n)+j}\right)\right) \tag{5.9e}$$

For simplicity the equations 5.9a, 5.9d and 5.9e are rewritten as:

$$\sum_{nmij} B^{i+j}k(\mathbf{x}_n,\mathbf{x}_m)(a_{Kn+i}a_{Km+j}-a_{Kn+i}a_{K(N+m)+j}-a_{K(N+n)+i}a_{Km+j}+a_{K(N+n)+i}a_{K(N+m)+j}) \tag{5.10a}$$

$$\xi\left(\sum_{nmij}(a_{Kn+i}a_{Km+j}-a_{Kn+i}a_{K(N+m)+j}-a_{K(N+n)+i}a_{Km+j}+a_{K(N+n)+i}a_{K(N+m)+j})\right) \tag{5.10b}$$

$$\beta\left(\sum_n\sum_i\sum_j B^{i+j}a_{Kn+i}a_{K(N+n)+j}\right) \tag{5.10c}$$

respectively. The cost function then becomes:

$$\begin{aligned}\sum_{n,m=0}^{N-1}\sum_{i,j=0}^{K-1}\Big(&a_{Kn+i}\tilde{Q}_{Kn+i,Km+j}a_{Km+j}-a_{Kn+i}\tilde{Q}_{Kn+i,K(n+m)+j}a_{K(N+m)+j}\\&-a_{K(N+n)+i}\tilde{Q}_{K(N+n)+i,Km+j}a_{Km+j}\\&+a_{K(N+n)+i}\tilde{Q}_{K(N+n)+i,K(N+m)+j}a_{K(N+m)+j}\Big)\end{aligned} \tag{5.11}$$

Here $\tilde{Q}$ is a $2NK\times 2NK$ matrix associated to the problem. Its elements are given by:

$$\tilde{Q}_{Kn+i,Km+j}=B^{i+j}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi)+\delta_{nm}\delta_{ij}B^i(\epsilon-t_n) \tag{5.12a}$$

$$\tilde{Q}_{Kn+i,K(N+m)+j}=-B^{i+j}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi-\beta) \tag{5.12b}$$

$$\tilde{Q}_{K(N+n)+i,Km+j}=-B^{i+j}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi) \tag{5.12c}$$

$$\tilde{Q}_{K(N+n)+i,K(N+m)+j}=B^{i+j}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi)+\delta_{nm}\delta_{ij}B^i(\epsilon+t_n) \tag{5.12d}$$

for $n,m\in\{0,\dots,N-1\}$ and $i,j\in\{0,\dots,K-1\}$. If the encoding is selected according to equations 5.4 and 5.5 the matrix takes as final formulation:

$$\tilde{Q}_{Kn+i,Km+j} = B^{i+j-2k_0}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi) + \delta_{nm}\delta_{ij}B^{i-k_0}(\epsilon - t_n) \tag{5.13a}$$

$$\tilde{Q}_{Kn+i,K(N+m)+j} = -B^{i+j-2k_0}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi-\beta) \tag{5.13b}$$

$$\tilde{Q}_{K(N+n)+i,Km+j} = -B^{i+j-2k_0}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi) \tag{5.13c}$$

$$\tilde{Q}_{K(N+n)+i,K(N+m)+j} = B^{i+j-2k_0}(\frac{1}{2}k(\mathbf{x}_n,\mathbf{x}_m)+\xi) + \delta_{nm}\delta_{ij}B^{i-k_0}(\epsilon + t_n) \tag{5.13d}$$

By construction the matrix is symmetric therefore it is possible to obtain the upper-triangular matrix Q associated to the QUBO problem by considering:

$$Q_{i,j} = \begin{cases} \tilde{Q}_{i,j} + \tilde{Q}_{j,i}, & \text{if } i < j; \\ \tilde{Q}_{i,j}, & \text{if } i = j; \\ 0, & \text{otherwise} \end{cases} \tag{5.14}$$

5.3 Combination of the solutions

The D-wave quantum annealer provides as output a set of solutions whose number is selected by the user. For the experimental validation conducted in this work 40 solutions were considered for each problem instance. Some solutions might be better than others therefore it is of utmost importance to combine the solutions giving more credit to the best solutions. To do so a dataset called *validation set* is used to determine which solutions for the coefficients $\underline{\alpha}$ and $\underline{\hat{\alpha}}$ are better. The dataset consists of samples made by couples of input output values $(\mathbf{x}_n, y_n)$, $\quad n \in \{0,\dots,V-1\}$, where V is the dimensionality of the validation set. The input $\mathbf{x}$ can in general be a vector of a given dimension but in our test cases we are focusing only on one-variable function. Each solution provided by the annealer is denoted as $(\underline{\alpha}, \underline{\hat{\alpha}})_i$ where the subscript i denotes the i-th solution provide as output by the annealer. It might be useful to visualize the values of each solutions as elements of matrix S of size $M \times 2N$, where M is the number of solutions provided by the annealer. Each row of the matrix corresponds to a candidate solution. Within any row the first N elements of the matrix correspond to the coefficients $\underline{\alpha}$ whereas the remaining N correspond to the coefficients $\underline{\hat{\alpha}}$. The matrix takes the form:

$$S = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdot & \cdot & \alpha_{1N} & \hat{\alpha}_{11} & \cdot & \cdot & \hat{\alpha}_{1N} \\ \alpha_{21} & \alpha_{22} & \cdot & \cdot & \alpha_{2N} & \hat{\alpha}_{21} & \cdot & \cdot & \hat{\alpha}_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{(M-1)1} & \alpha_{(M-1)2} & \cdot & \cdot & \alpha_{(M-1)N} & \hat{\alpha}_{(M-1)1} & \cdot & \cdot & \hat{\alpha}_{(M-1)N} \\ \alpha_{M1} & \alpha_{M2} & \cdot & \cdot & \alpha_{MN} & \hat{\alpha}_{M1} & \cdot & \cdot & \hat{\alpha}_{MN} \end{bmatrix} \tag{5.15}$$

For each solution a prediction on the validation set is made by substituting the values of $(\underline{\alpha}, \underline{\hat{\alpha}})_i$ into equation 3.31. To assign credit to each solution a cost function between the actual and the predicted values is calculated. Two cost function are considered in our analysis: the mean squared error (MSE) and the log-cosh loss, whose formulae are:

$$\text{MSE} = (Y^{true}, Y^{predicted}) = \frac{1}{N}\sum_{i=1}^{N}(y_i^{true} - y_i^{predicted})^2 \tag{5.16}$$

$$\text{log-cosh loss}(Y^{true}, Y^{predicted}) = \sum_{i=1}^{N}\log_2(\cosh(y_i^{true} - y_i^{predicted})) \tag{5.17}$$

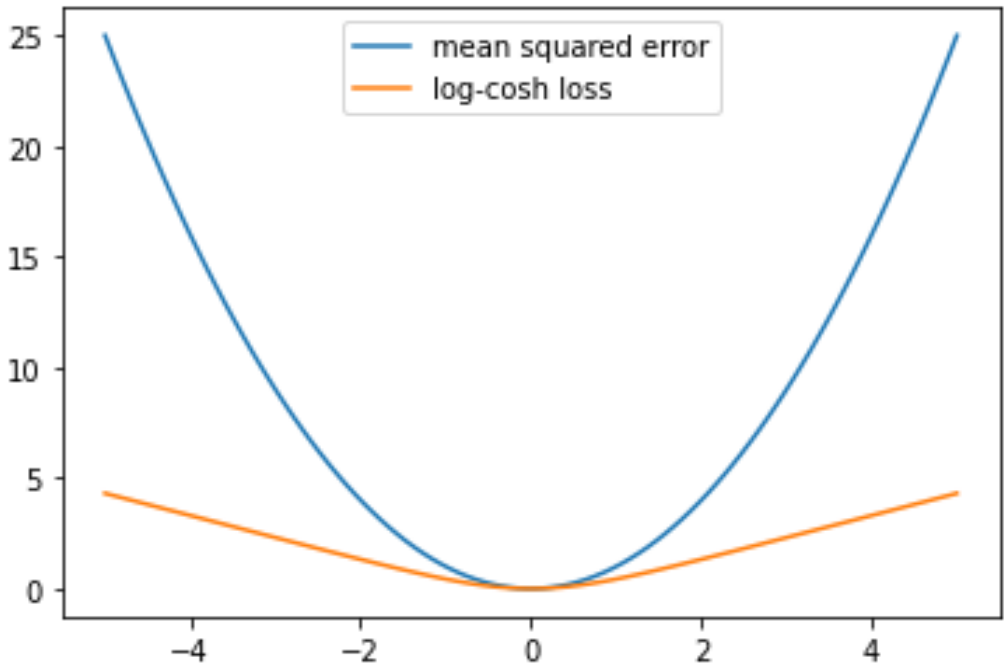


Figure 5.1: Comparison between MSE and log-cosh loss function

As can be seen in figure 5.1 the behaviour of the two functions is very similar for small values of the error, whereas for bigger ones the log-cosh tend to give less penalty. In this work 2 has been used as value for the base of the logarithm but others can be used. By using the log-cosh therefore the cost function will penalize less outliers. Every solution is therefore associated with 2 error measure that are used to assign credit, which is calculated by evaluating the multiplicative inverse of the errors. These quantities will be reffered to as *mse score* and *lc score*. The *mse score* and *lc score* can be interpreted as vectors of dimension M, where the i-th component corresponds to a measure of how well the solution $(\underline{\alpha}, \hat{\underline{\alpha}})_i$ performed on the prediction of the validation set. The elements of the score vectors will be used as coefficients of a weighted average for the $\underline{\alpha}$ and $\hat{\underline{\alpha}}$. To do so, however, the entries of the vectors need to satisfy the following properties: each one must be non-negative and their sum must be equal to 1. The first condition is always satisfied by the choice of the loss functions, whereas to ensure the second condition two different methodologies were proposed. In the first one, that is referred to as *scores normalization* each element of the vector is divided by th sum of all the elements of the vector, while in the other methodology a softmax is applied to the vector. The softmax function $\sigma : \mathbb{R}^d \to \mathbb{R}^d$, which is widely used in machine learning, is defined as follows:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \tag{5.18}$$
$$\text{with } \mathbf{z}, \sigma(\mathbf{z}) \in \mathbb{R}^d \text{ and } i \in \{1, \dots, d\}$$

Both the scores normalization and the softmax ensure that the entries of the score vectors can be interpreted as the coefficients of a convex combination that is then used to calculate the weighted average. Specifically, 4 different weighted average can be implemented, by combining the choice of the cost function and the choice of the strategy to obtain the coefficient of the convex combination. In the experimental analysis will be also considered as a method to obtain a final solution the process of simply selecting the best set of $\underline{\alpha}$ and $\hat{\underline{\alpha}}$ according to the performances on the validation set. For each method the procedure to obatin the final solution, given the vector of the weight coefficients $\mathbf{w}$ of dimension M, is the same. Given the matrix S described in equation 5.15 the vector $(\underline{\alpha}, \hat{\underline{\alpha}})^{final}$ of dimension 2N is obtained with:

$$(\underline{\alpha}, \hat{\underline{\alpha}})_i^{final} = \sum_{j=1}^{M} w_j S_{ji} \text{ with } i \in \{1, \dots, 2N\} \tag{5.19}$$

Therefore the 5 methods for combining the solution will be:

- *scores normalization or scores norm*: it uses as loss function the mean square error and each coefficient is obtained by dividing itself by the sum of all the others
- *softmax*: it uses the mean square error as cost function and the softmax function to get the coefficients of the weighted average
- *lc scores norm* it uses the log-cosh as cost function and calculate the weights by dividing each score by the sum of all the others
- *lc softmax* it uses log-cosh as cost function and then uses sofmax function to get the coefficients
- *best set of alphas*: returns the best set of $\underline{\alpha}$ and $\underline{\hat{\alpha}}$ based on wchich set of such values achieved the smallest error on the validation set considering as loss function the mean squared error

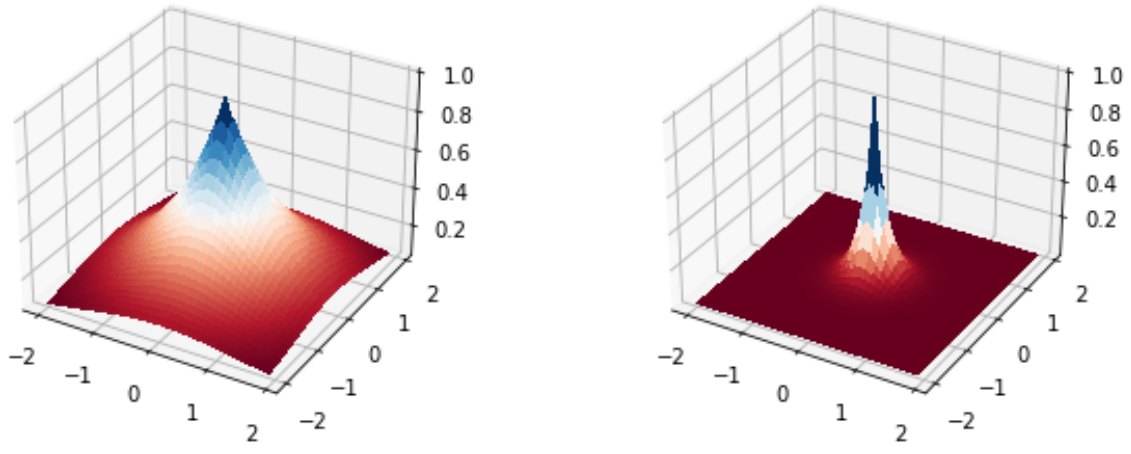
Chapter 6

Applications and test cases

The results of applying the quantum SVR to the approximation of different target functions will now be shown. In the proposed examples the implementation the quantum SVR is applied to very simple problems where the train and the test set consist of samples from a one-variable function. Specifically, 3 different functions are considered: a sine function a sinc and a triangular function. For each function two different cases, for a total of six different test cases, are analysed: in the first one the training set is the same for each run and consist of samples equally spaced in the input domain, whereas for the latter the training set is different for each iteration and is randomly sampled. To make the comparison between the traditional and the quantum SVR fair the validation set for the combination of the annealing solutions will be the same dataset used for the training phase, therefore both traditional and quantum SVR will use the same number of samples for the training phase. In fact, using additional training sample only for the quantum case might alter the results of the comparison. In the experimental validation all the results of the different implementations of the QSVR are compared with the traditional one on a test set that consists of 100 datapoints randomly taken from the considered input range. For each of the six different test cases the results of 7 experiments, chosen from all experiments that have been carried out, are illustrated and commented. Both implementations will employ the same radial basis function (RBF) kernel. Such kernels have the property that they depend only on the distance of the input, *i.e* $K(\mathbf{x}_n, \mathbf{x}_m) = K(\|\mathbf{x}_n - \mathbf{x}_m\|)$. The kernel that will be used in our experimental analysis is the gaussian kernel:

$$K(\mathbf{x}_n, \mathbf{x}_m) = e^{-\gamma\|\mathbf{x}_n - \mathbf{x}_m\|} \tag{6.1}$$

where γ is an hyperparameter.



(a) $\gamma = 1$ (b) $\gamma = 5$

Figure 6.1: Graphical depiction of the gaussian RBF kernel with different values of γ

The values for the hyperparameters γ, C and ϵ, will be the same for both SVR and QSVR. Moreover, for both of them a high value of the hyperparameter C will be used because we are interested in the study of the overfitting case. The choice of the value C will also be chosen in such a way that the constraints 3.29 and 3.30 are satisfied, i.e choosing a value for C that is higher to the quantity defined in equation 5.6.
For the implementation of the traditional SVR as well as the generation of the dataset the Python library Scikit-learn was used whereas the QSVR was implemented using the D-wave Ocean software. For the generation of the plots and tables the library Matplotlib was used. For each of the considered test cases the number of training samples used was 20. The number was kept the same for each of the test functions in order to assess the performances of the QSVR in different situations with the same training resources.

6.1 Test case: sine function

In this test case the quantum SVR was applied to the estimation of a sine function. The quantum SVR and the traditional SVR are tested on a dataset that consists of samples from a sine function: each training samples is therefore made up by the tuple $(x, A\sin(fx))$, where A denotes the amplitude of the oscillation and f its frequency. For this setting the values for A and f were 1 and 2, respectively, while the input range of the independent variable was $[0, 5]$. The value for the hyperparameter C is set to 1000 beacuse we are interested in the study of the overfitting case.

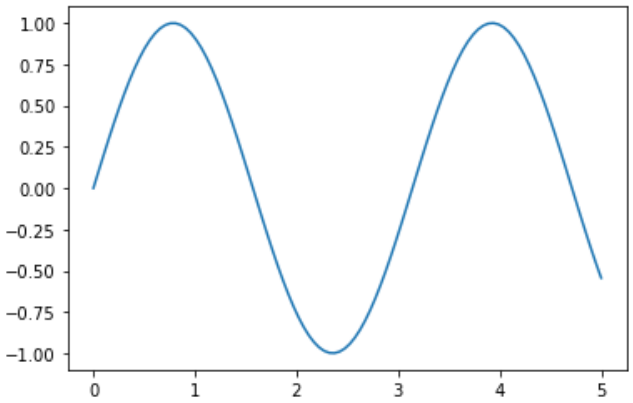


Figure 6.2: the function $y = \sin(2x)$ in the considered range for x

6.1.1 First test case: evenly spaced training points in the input domain

Some examples of the application of the QSVR to the approximation of the function $y = \sin(2x)$ will be now illustrated and commented.

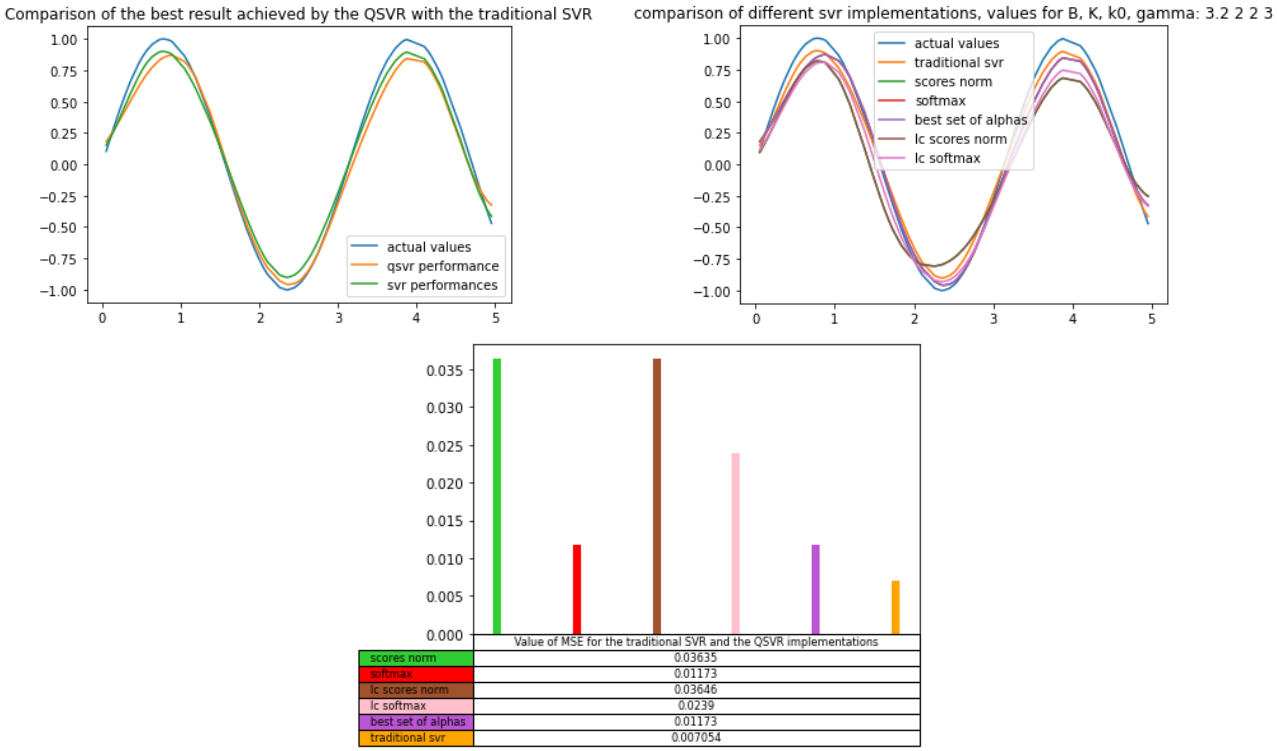


Figure 6.3: Results for test 1 for the function $y = \sin(2x)$

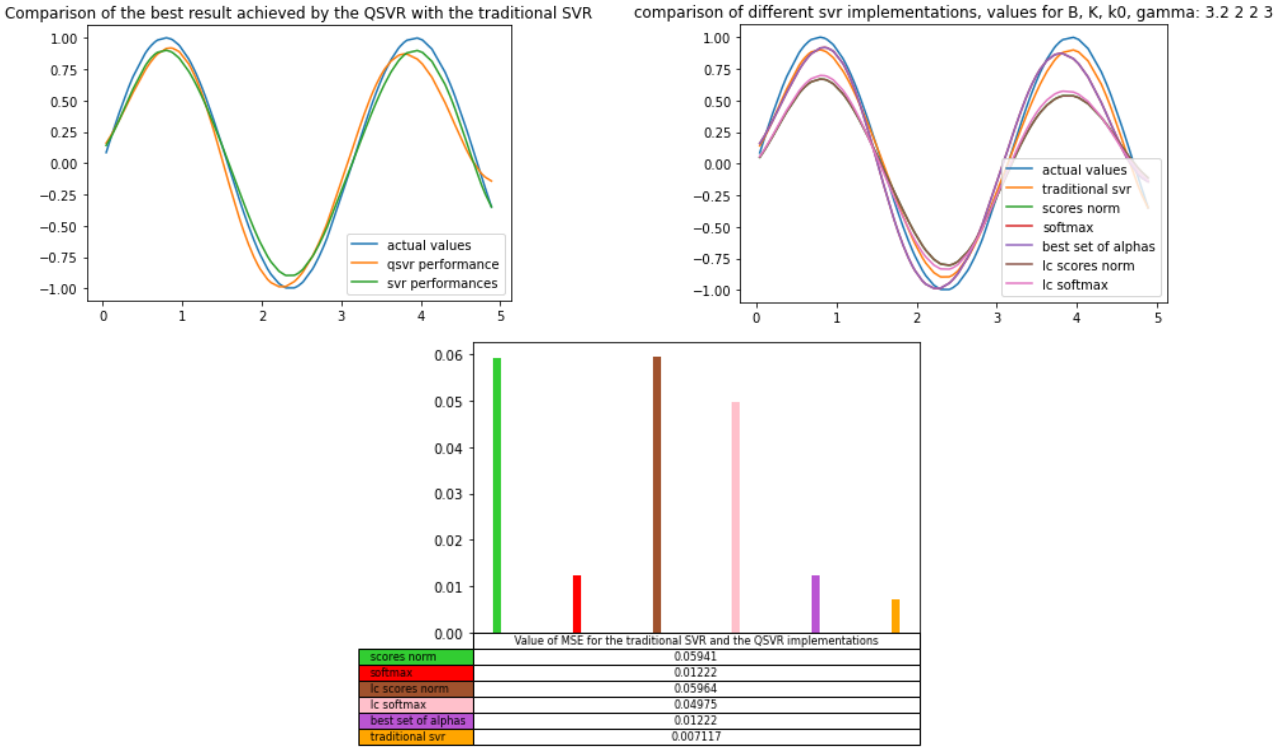


Figure 6.4: Results for test 2 for the function $y = \sin(2x)$

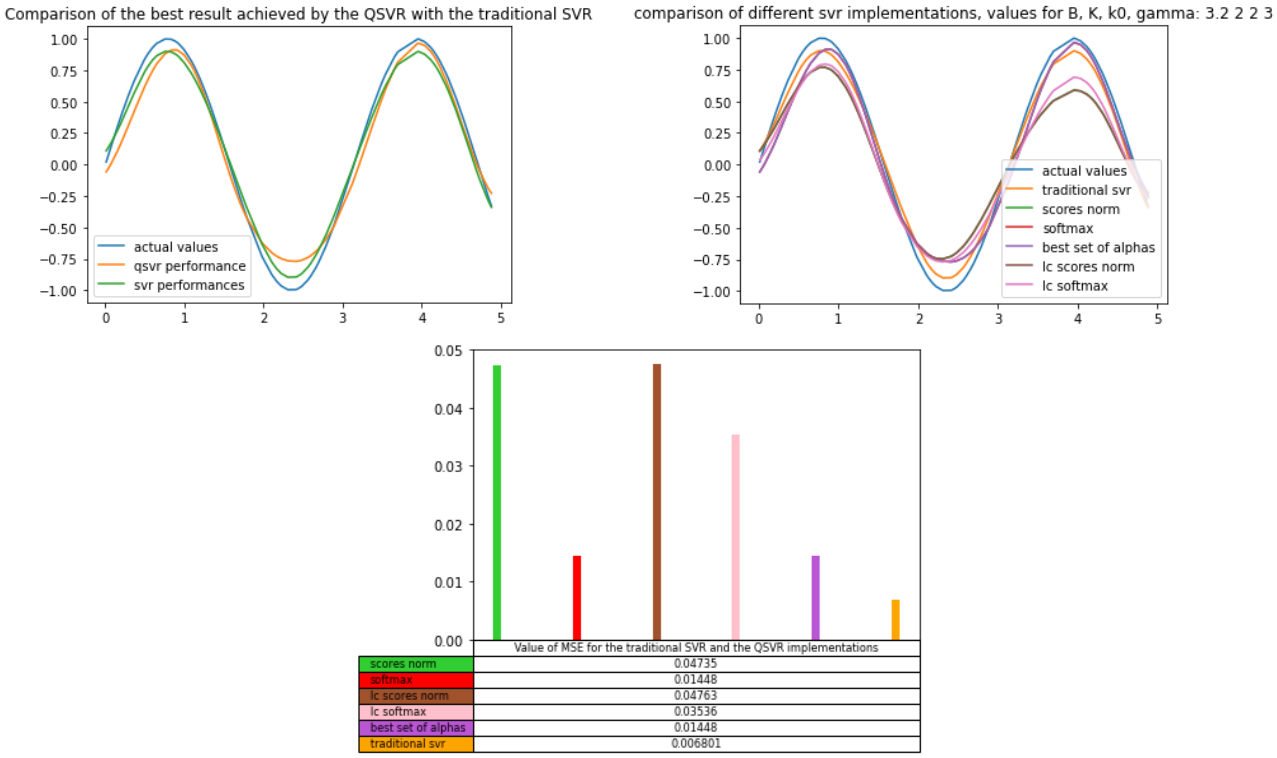


Figure 6.5: Results for test 3 for the function $y = \sin(2x)$

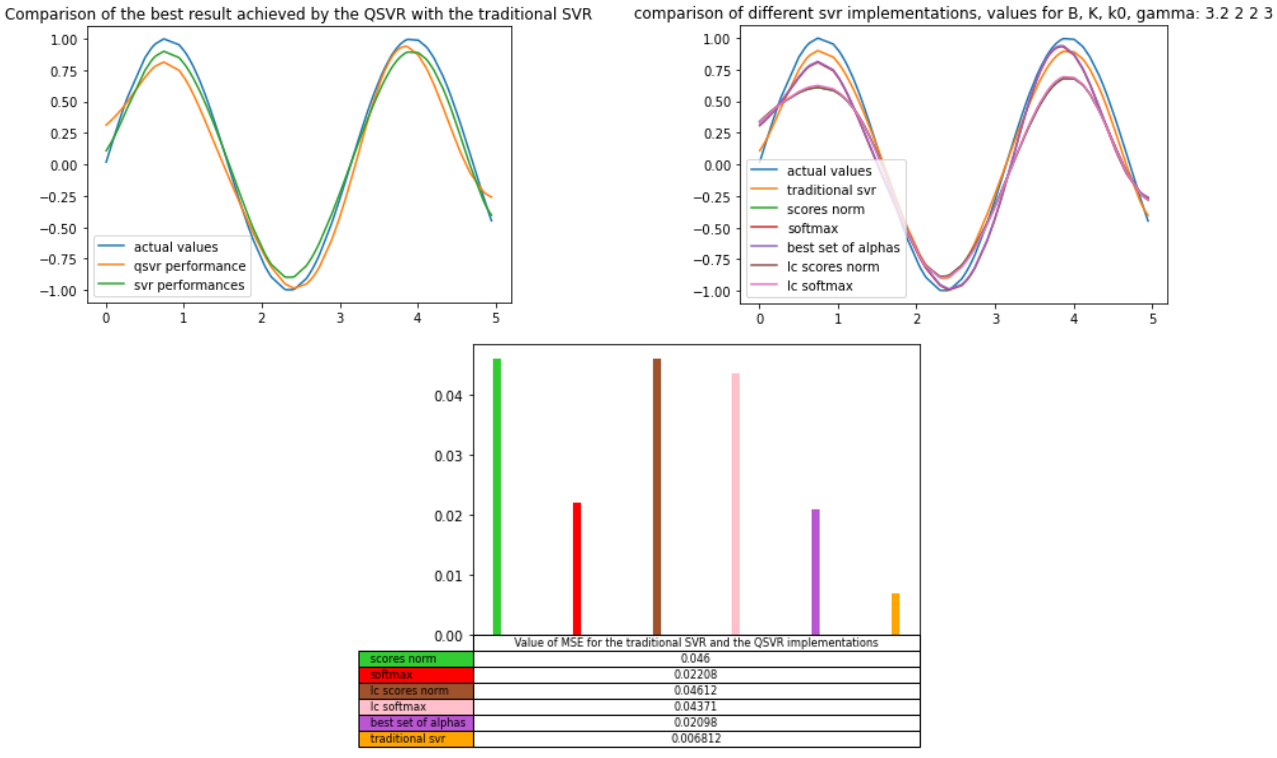


Figure 6.6: Results for test 4 for the function $y = \sin(2x)$

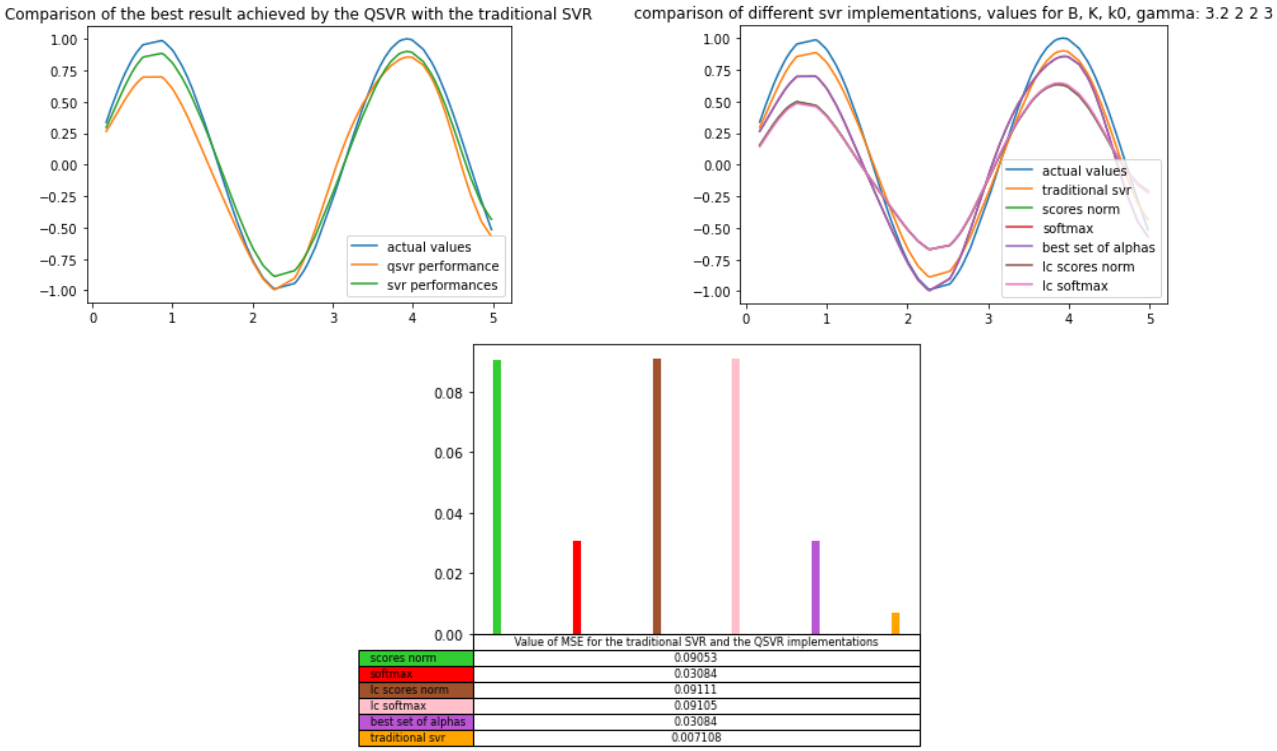


Figure 6.7: Results for test 5 for the function $y = \sin(2x)$

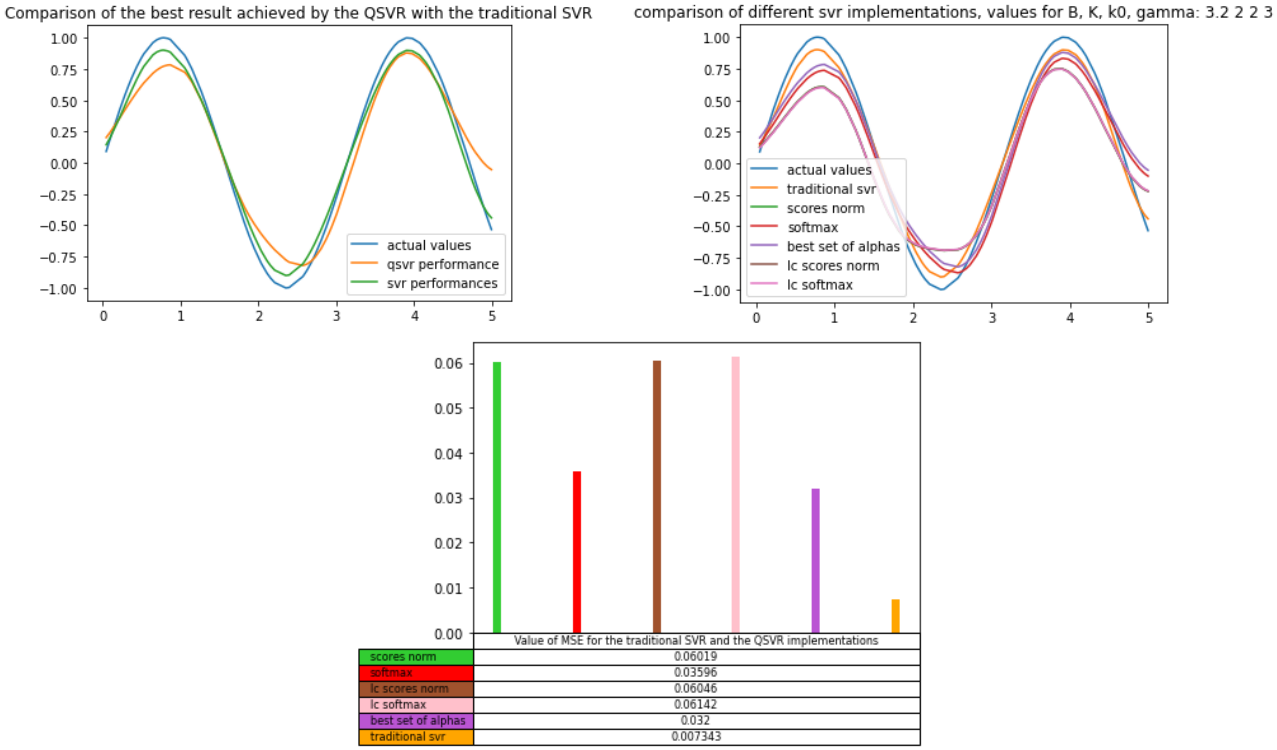


Figure 6.8: Results for test 6 for the function $y = \sin(2x)$

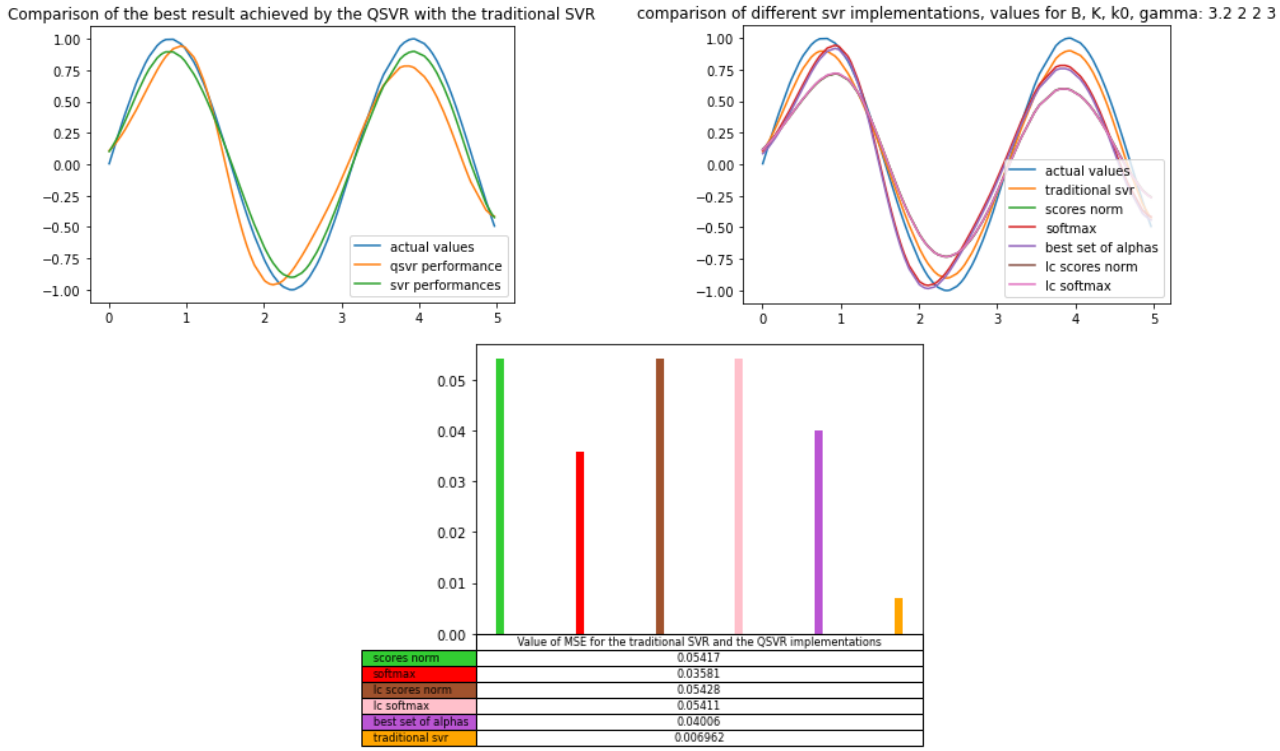


Figure 6.9: Results for test 7 for the function $y = \sin(2x)$

The images above refer to 7 tests that were run on a training dataset that consisted of 20 samples evenly spaced in the input domain. The values for the hyperparameter of the QSVR were $B = 3.2$, $K = 2$, and $k_0 = 2$. Such values were found empirically by running some validation tests. In the various tests the different implementations of the QSVR performed differently with the methodology based on softmax and the one that consiedered only the best solution being the best. This fact suggests that focusing heavily on the best solutions and ignoring or giving small credit to the others might provide better results. This is also supported by the fact that the methodologies *scores norm*, *lc scores norm* and *lc softmax*, which tend to be less "elitary" and penalize less outliers, performed worse. As can be seen in the images above the traditional SVR managed to outperform the QSVR implementation in all tests, however the QSVR managed to achieve good results in approximating the target function. In particluar one fact worth noting is that the QSVR managed to infer the periodical behaviour of the function correctly with results comparable to those of the traditional SVR, but in contrast it performed worse when determining the values of the peaks and the valleys of the sine. This characteristic of the QSVR might be due to the fact that the through the encoding defined in 5.2 and 5.3 the values of $\underline{\alpha}$ and $\underline{\hat{\alpha}}$ are discretized.

6.1.2 Second test case: randomly sampled dataset

Some results on the same function with the same hyperparameters setting will now be presented. The difference is, however, that the training set used now considered samples taken randomly from the input range instead of taking them evenly spaced. In this section the same implementations of SVR and QSVR are now tested on a dataset that is randomly generated and varies for each experiments. The values of the hyperparameters are kept the same as before.

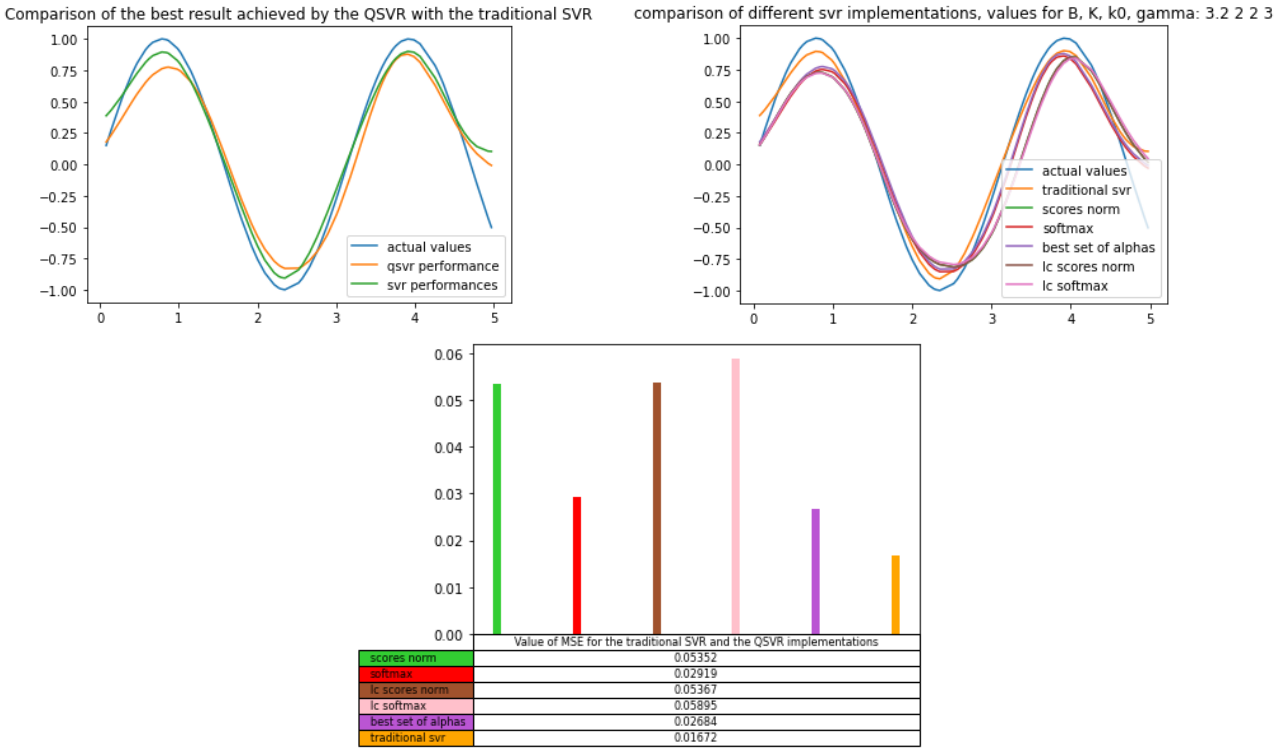


Figure 6.10: Results for test 1 for the function $y = \sin(2x)$

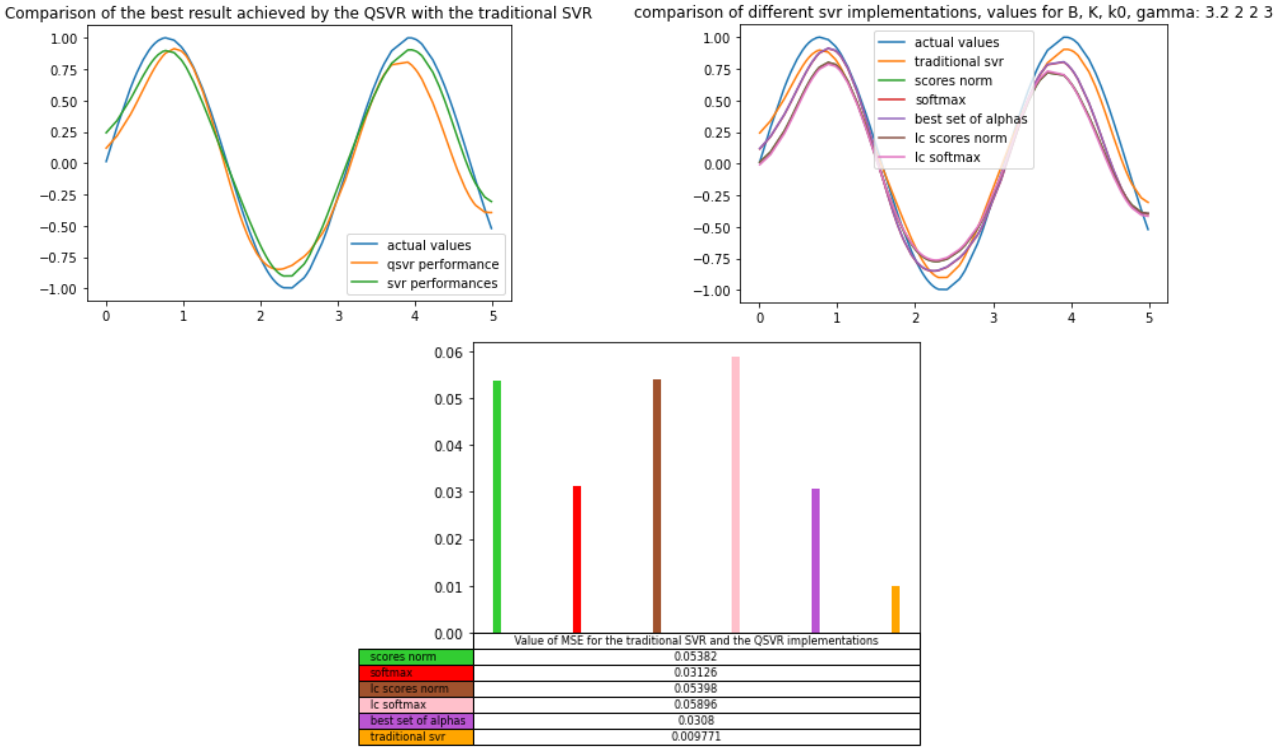


Figure 6.11: Results for test 2 for the function $y = \sin(2x)$

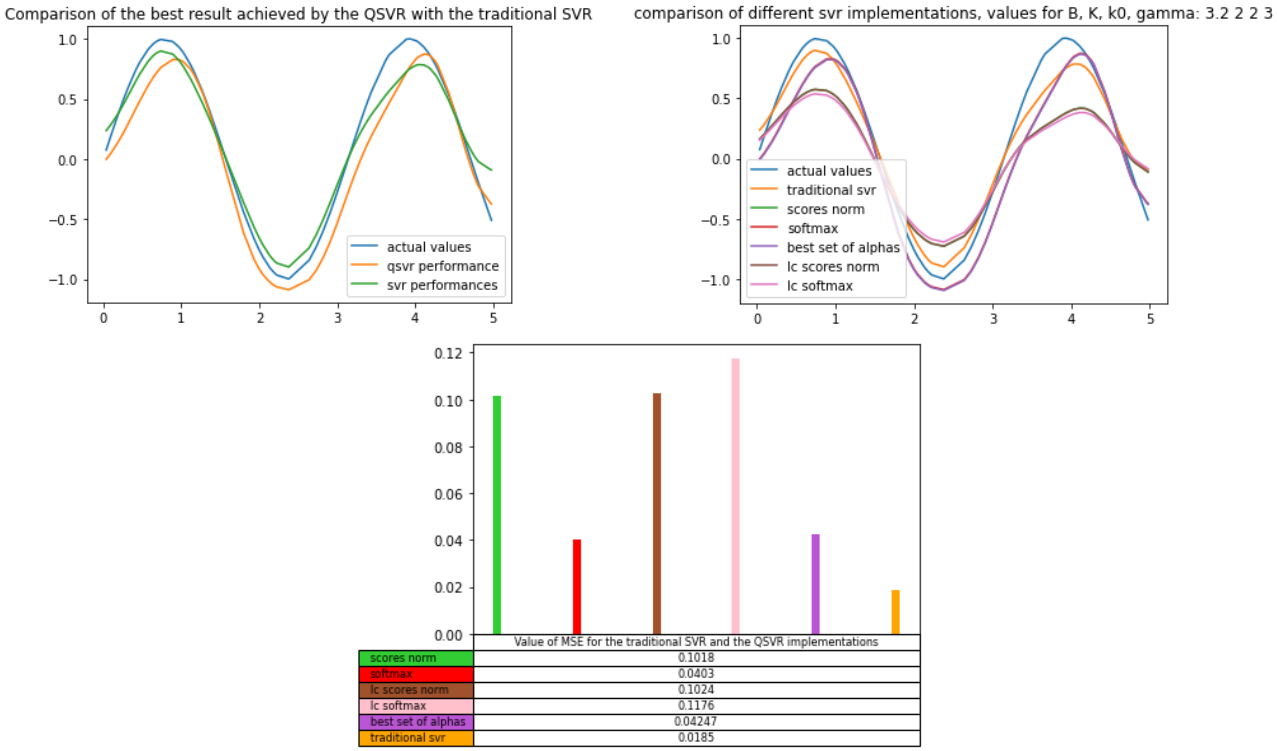


Figure 6.12: Results for test 3 for the function $y = \sin(2x)$

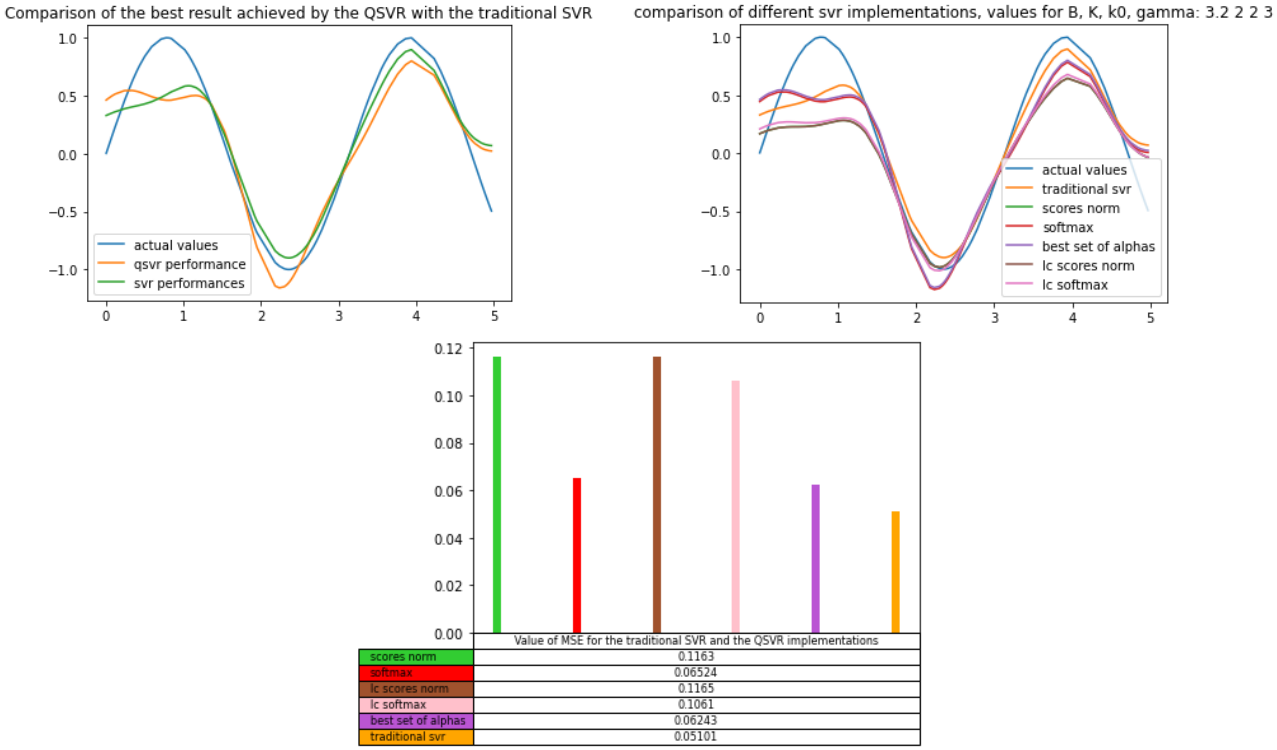


Figure 6.13: Results for test 4 for the function $y = \sin(2x)$

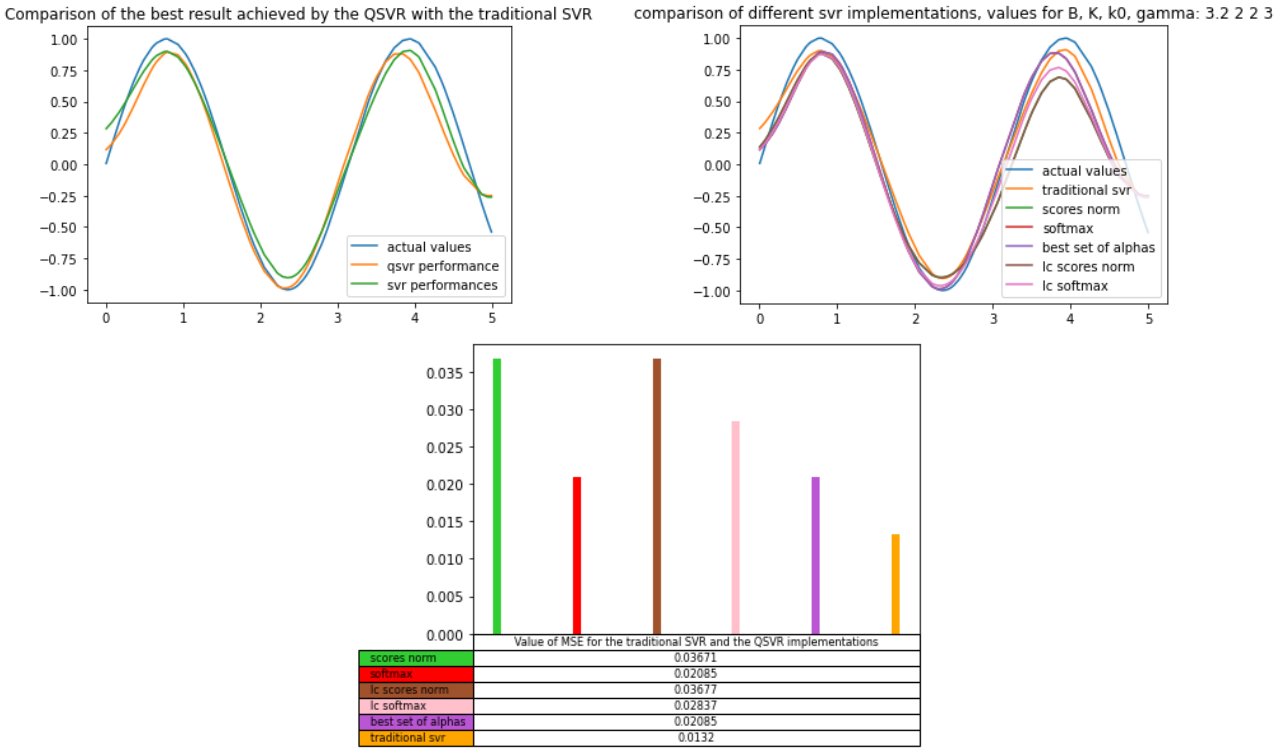


Figure 6.14: Results for test 5 for the function $y = \sin(2x)$

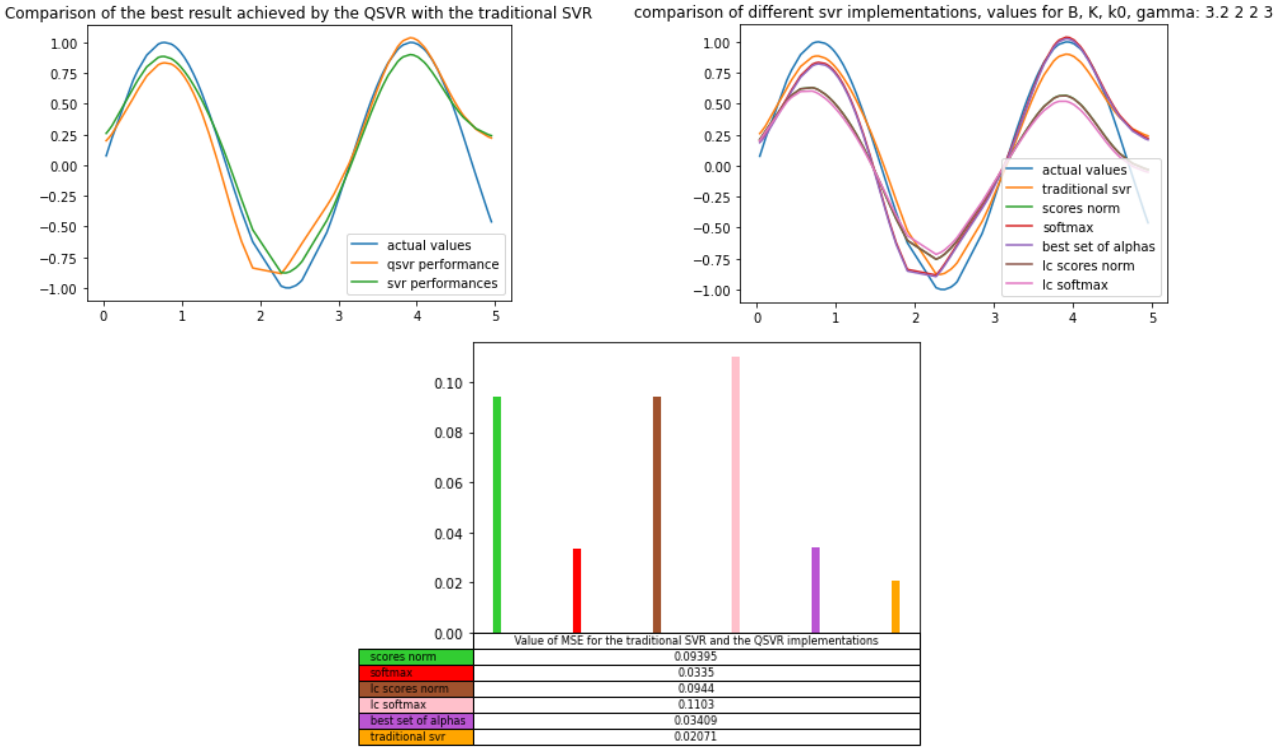


Figure 6.15: Results for test 6 for the function $y = \sin(2x)$

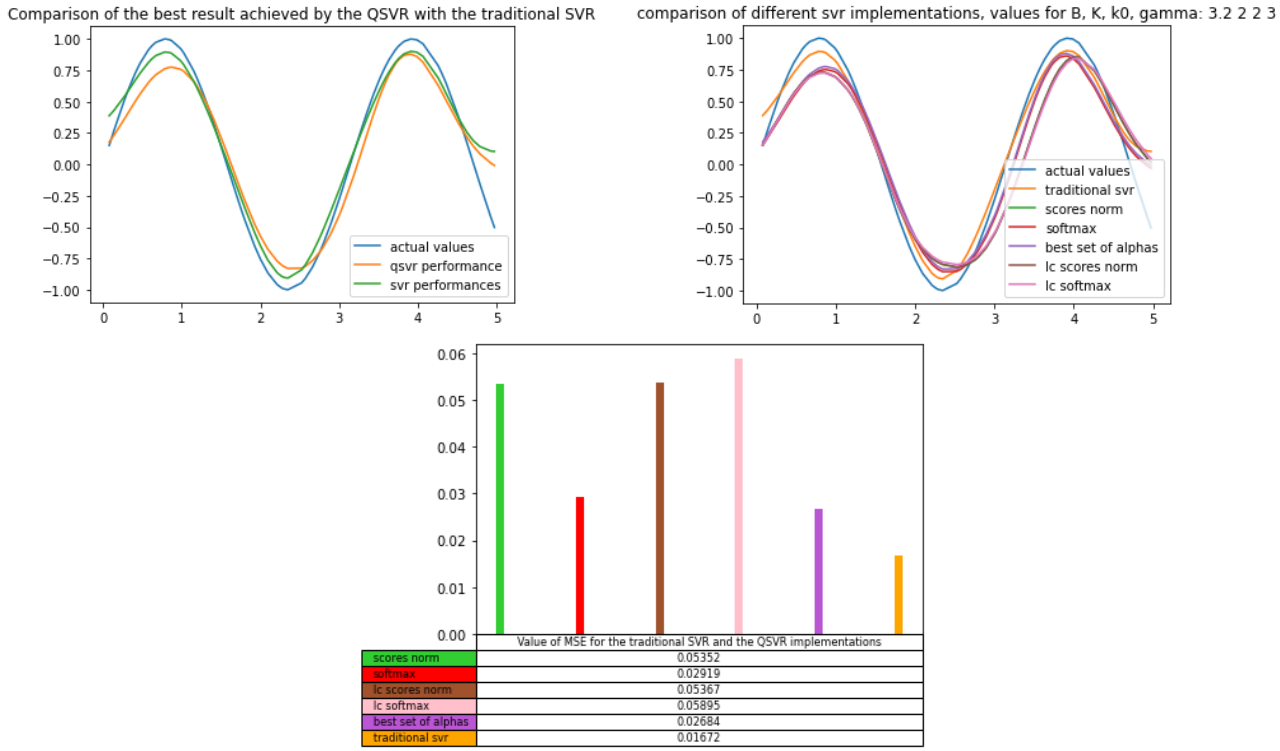


Figure 6.16: Results for test 7 for the function $y = \sin(2x)$

In this setting where the samples are now randomly taken from in the input range instead of being evenly spaced the QSVR achieved a similar performances with respect to the previous setting. In fact, the comparison between the scores achieved by the different implementations of QSVR and the performances of SVR are similar to the previous case. Figure 6.13 constitutes an interesting case: both classical and quantum SVR performed poorly in the input range around the first peak. However, it is important to point out that since both model did not perform well this behaviour is very likely to be attributable to the algorithm of the SVR itself. In this work, in fact, it was not proposed a new model for regression but rather an innovative way to optimize a pre-existing regression algorithm. This fact is of great interest in the context of how the different paradigms of quantum computing are applied to machine learning, in fact, as was pointed out in [14] "Circuit-based quantum computers have been predominantly used to compute the prediction of a quantum machine learning model that can be trained classically", "while quantum annealers have been proposed to optimize classical models". Finally, as in the previous case the best methods for the combinations of solutions are the softamx based on square-loss and the best set of alphas method.

6.2 Test case: sinc function

In this section I am going to analyze some results regarding the application of the proposed QSVR to the estimation of a sinc function. Specifically, the QSVR was applied to the estimation of the function $y = 5\text{sinc}(x)$. As for the test case of the sine function the input range considered is $[0, 5]$. The value of γ is set to 3, the same used in the previous test case, whereas the values of the hyperparameters concerning the encoding of the QSVR were: B=2.5, K=2 and $k_0 = 1$.

The value for the hyperparameter C is still set to 1000.

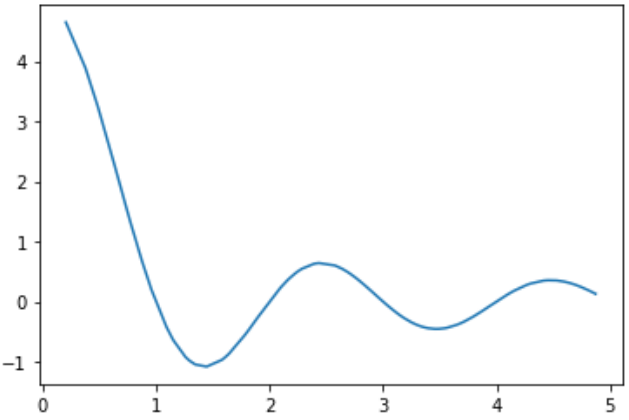


Figure 6.17: the function $y = 5\text{sinc}(x)$ in the considered range for x

6.2.1 First test case: evenly spaced training points in the input domain

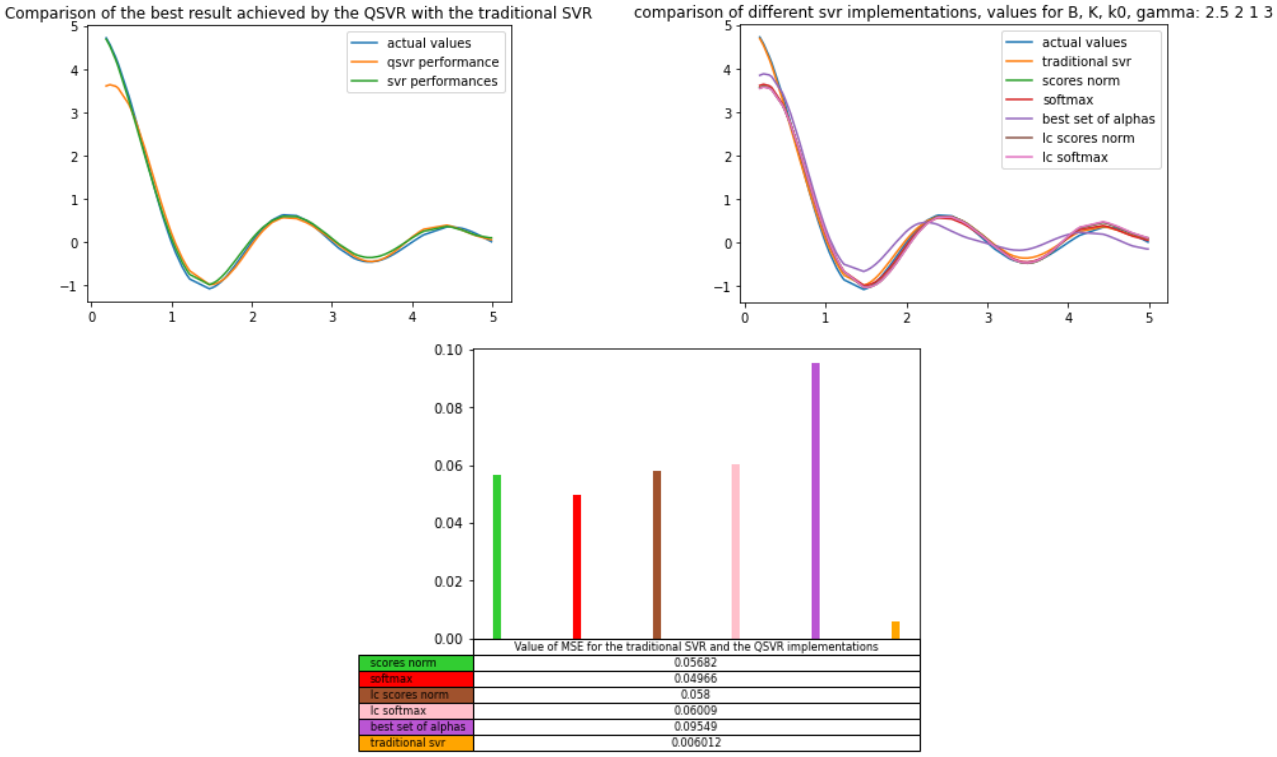


Figure 6.18: Results for test 1 for the function $y = 5\text{sinc}(x)$

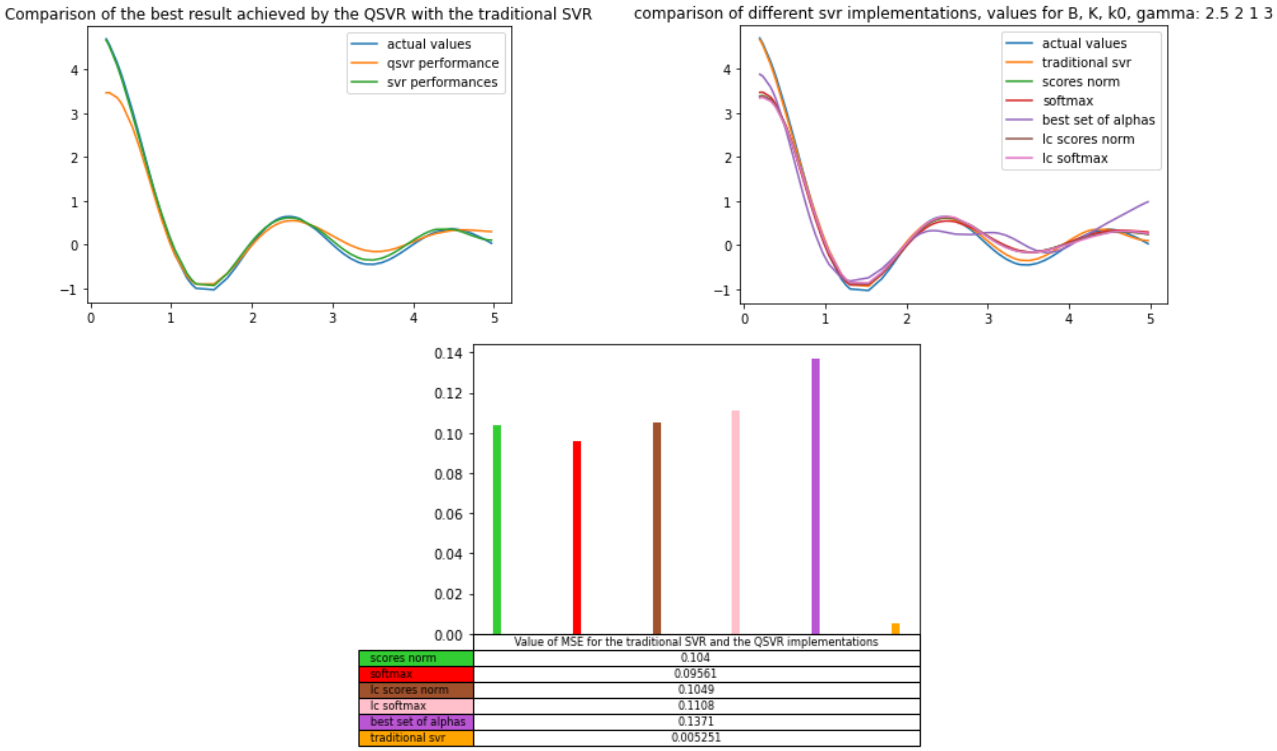


Figure 6.19: Results for test 2 for the function $y = 5\text{sinc}(x)$

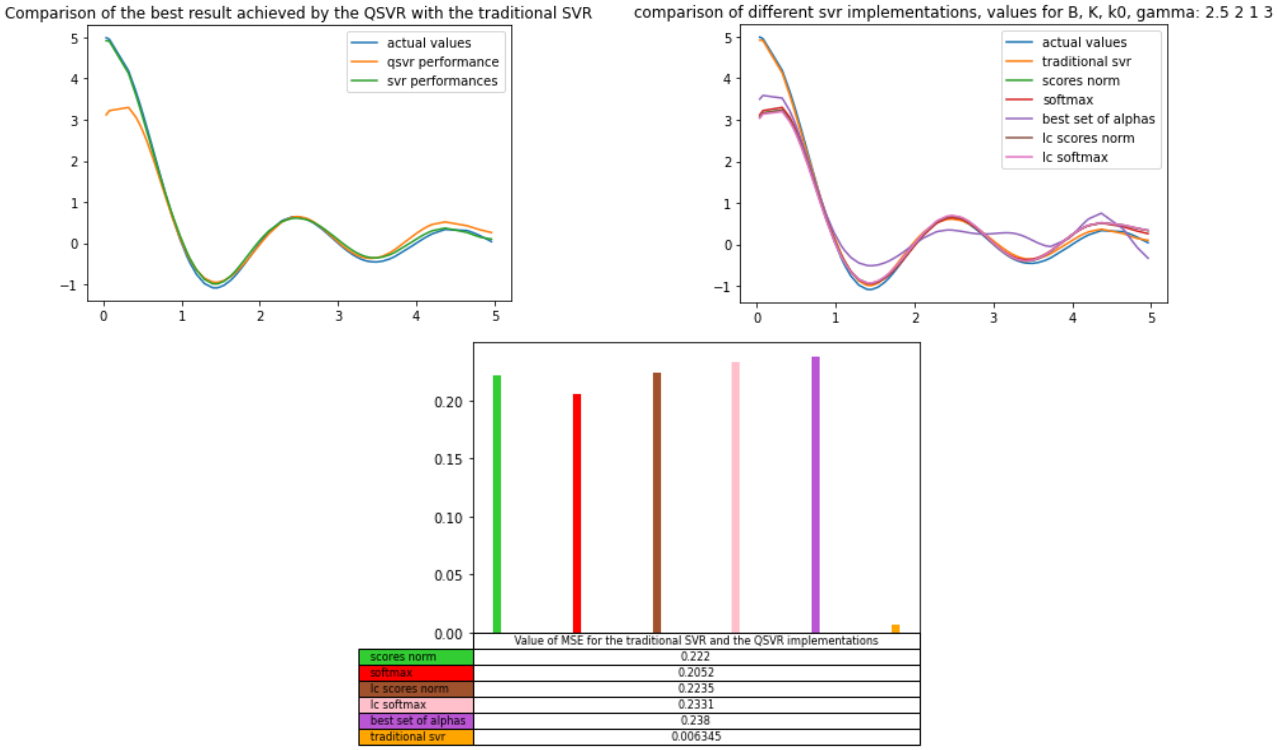


Figure 6.20: Results for test 3 for the function $y = 5\text{sinc}(x)$

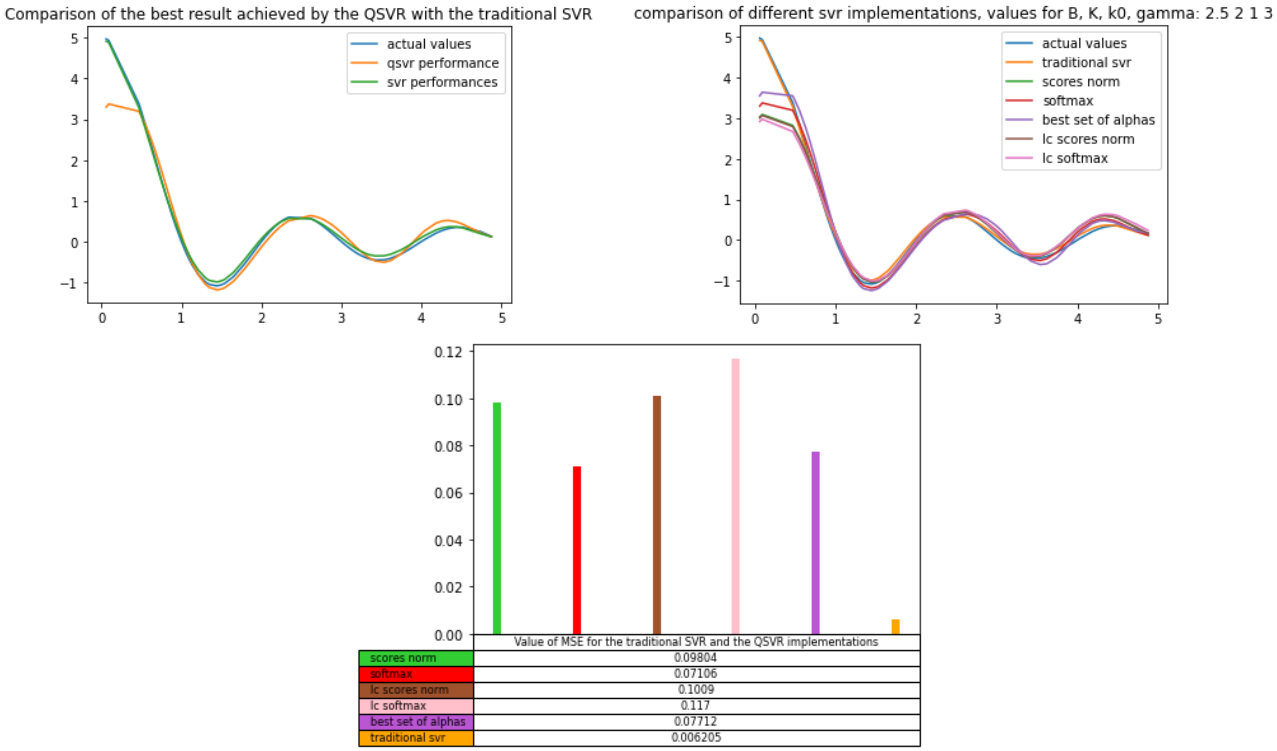


Figure 6.21: Results for test 4 for the function $y = 5\text{sinc}(x)$

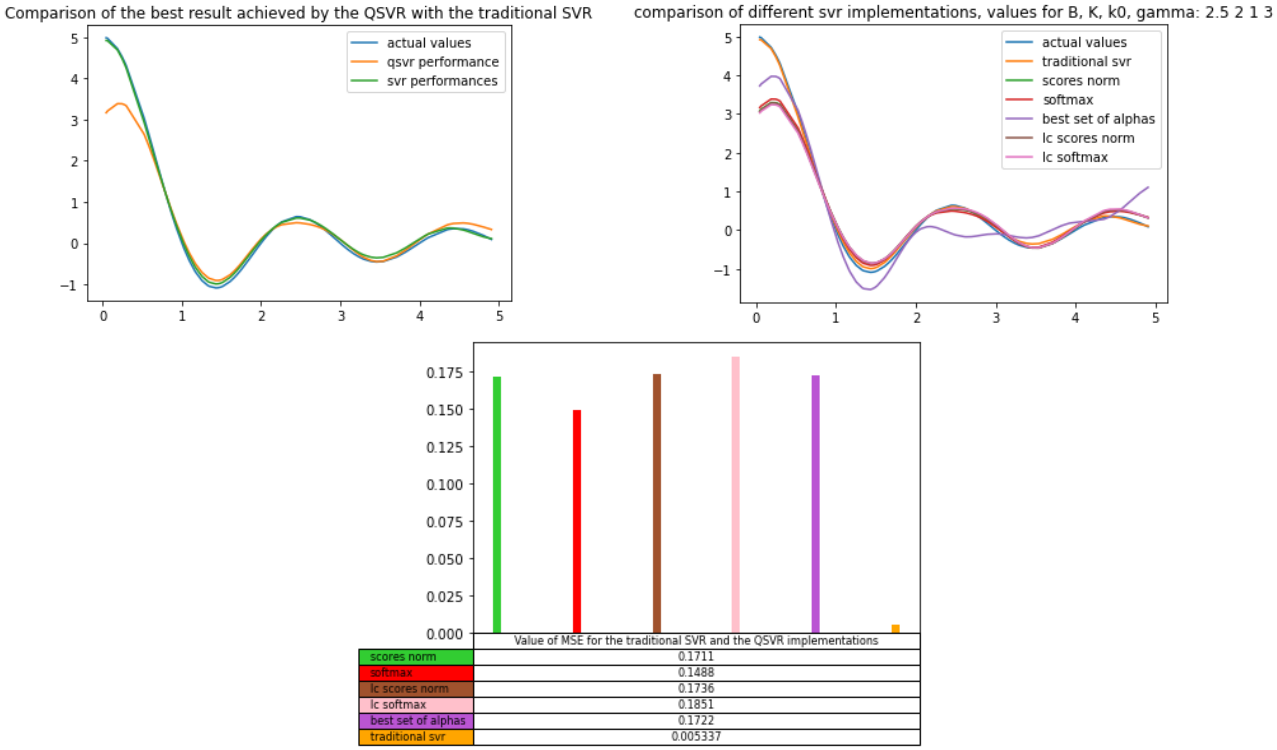


Figure 6.22: Results for test 5 for the function $y = 5\text{sinc}(x)$

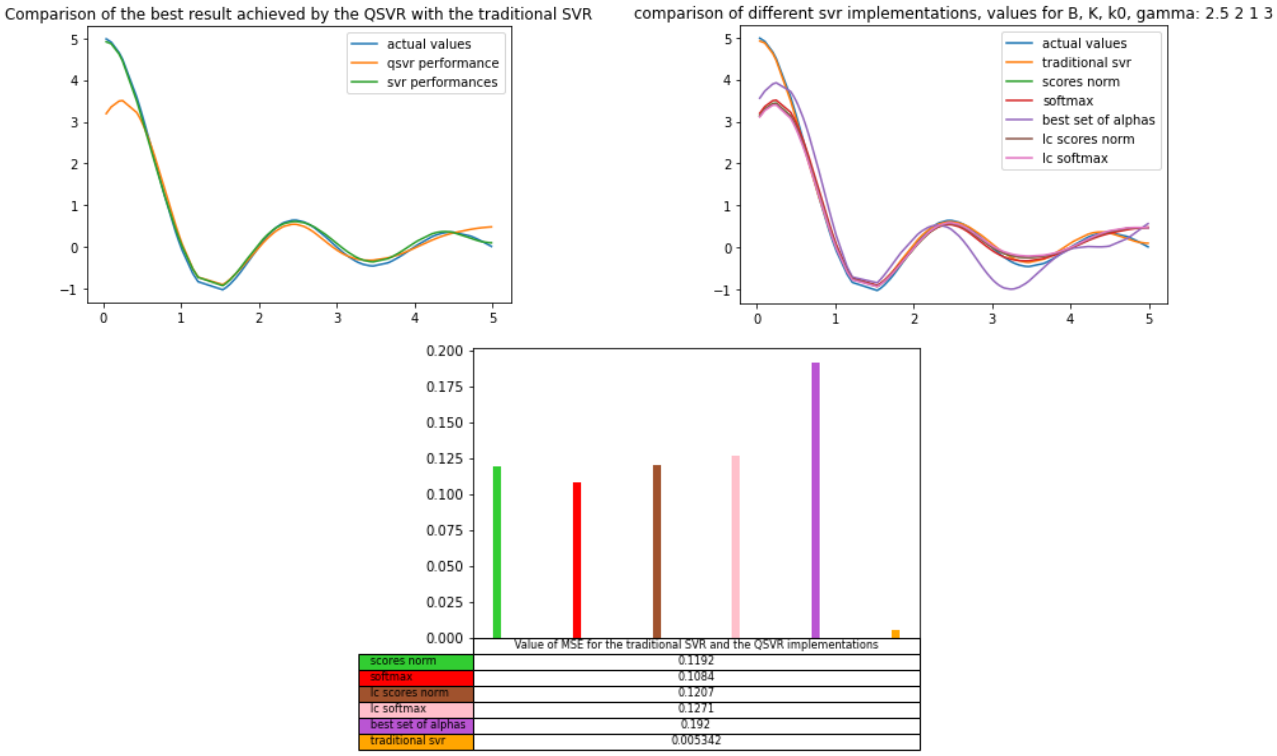


Figure 6.23: Results for test 6 for the function $y = 5\text{sinc}(x)$

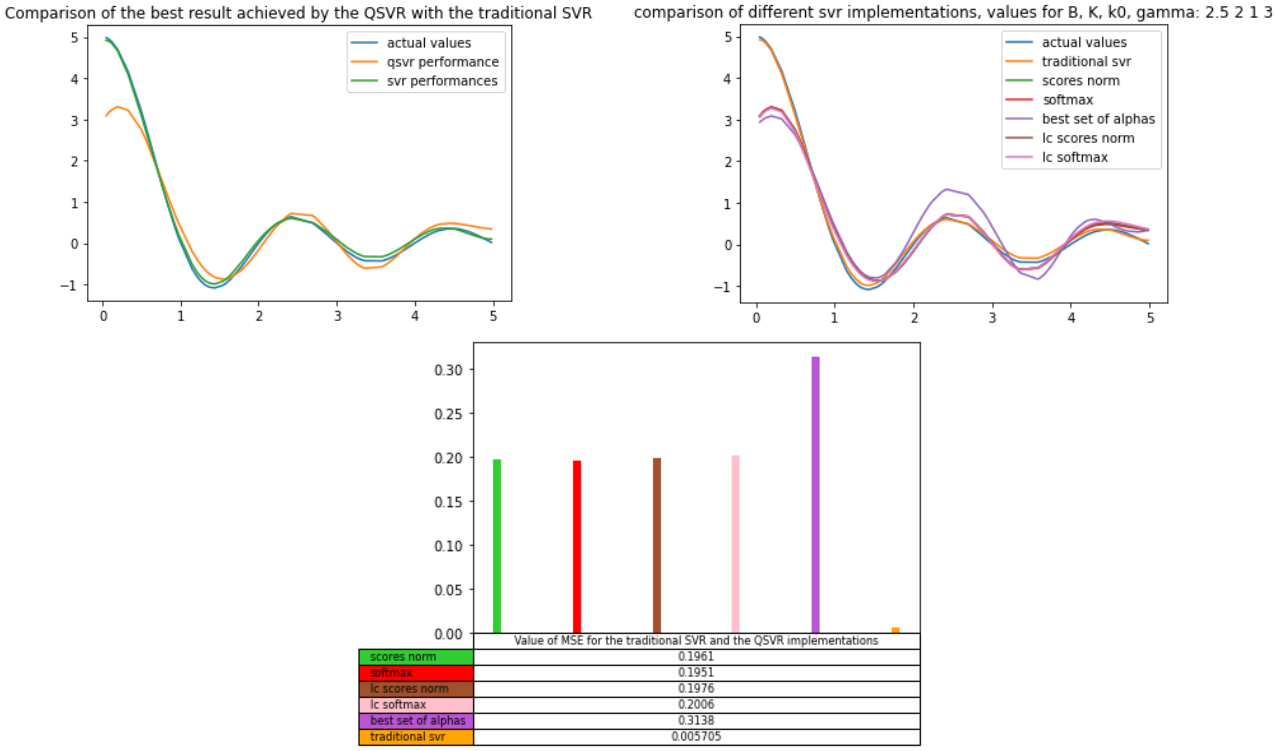


Figure 6.24: Results for test 7 for the function $y = 5\text{sinc}(x)$

The results achieved by the QSVR for the sinc function are worse in terms of mean squared error when compared with the previous test case. However, by observing the images it is possible to notice that the QSVR approximated the target function very well, almost equal to the classical implementation, in the whole input domain with the exception of the first part where the function has a higher slope. The reason for this is likely to be attributable to the aforementioned discretization of the α_n and

$\hat{\alpha}_n$, but further investigation is needed to confirm this hypothesis. For the validation phase aimed at determine the best values for the hyperparameters the QSVR was in fact tested on the same function with a higher value for K, and thus a higher number of possible values for the α_n and $\hat{alpha}_n$, but such configuration led to worse results. Another interesting result that can be observed in this test-case is how the different implementations of the QSVR performed in terms of accuracy on the test set. In the experiments conducted on the sine function the implementations *softmax* and *best set of alphas* always obtained better accuracy score than the others. For the sinc function, however, this is no longer the case and the various implementations performed vaery differently throughout the different test-runs. This is particularly evident in figure 6.18 and 6.19 where the method best set of alphas was the one that obtained the worst accuracy score for the estimation of the test set. This indicates that the performances of a particular method for the combination of the solutions depends on the considered function.

6.2.2 Second test case: randomly sampled dataset

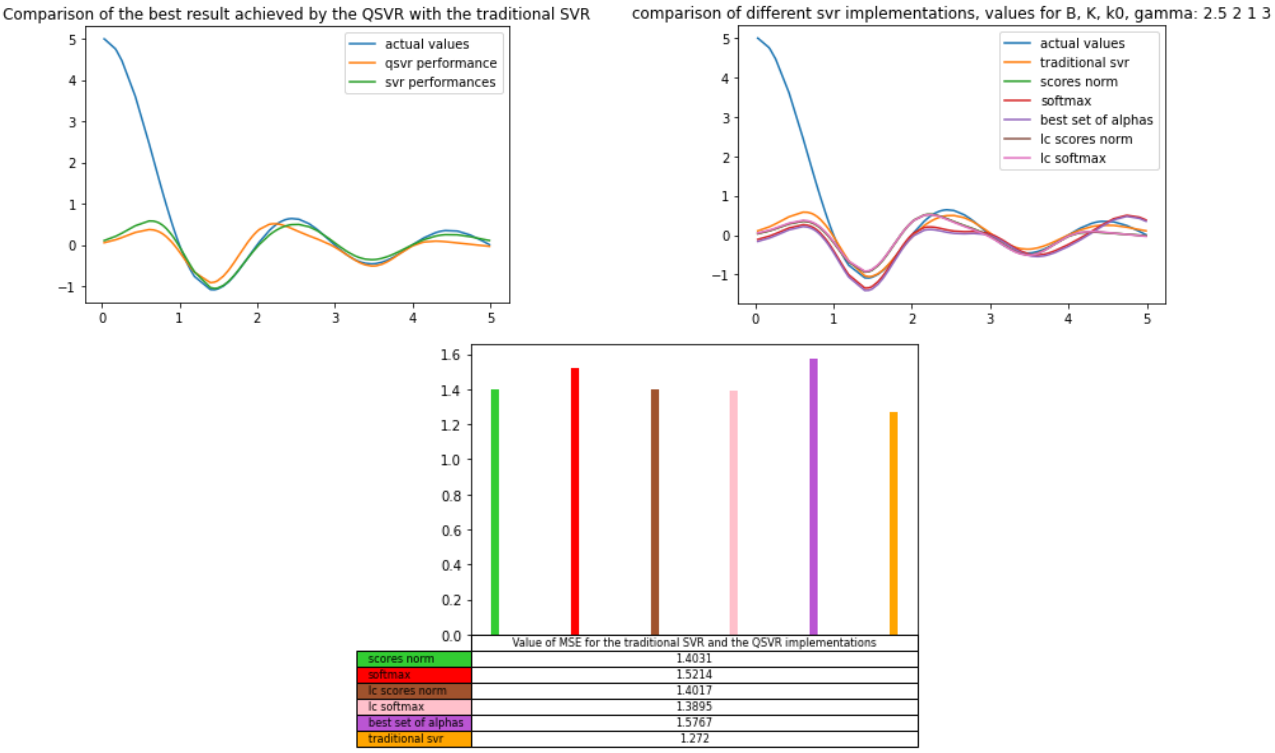


Figure 6.25: Results for test 1 for the function $y = 5\text{sinc}(x)$

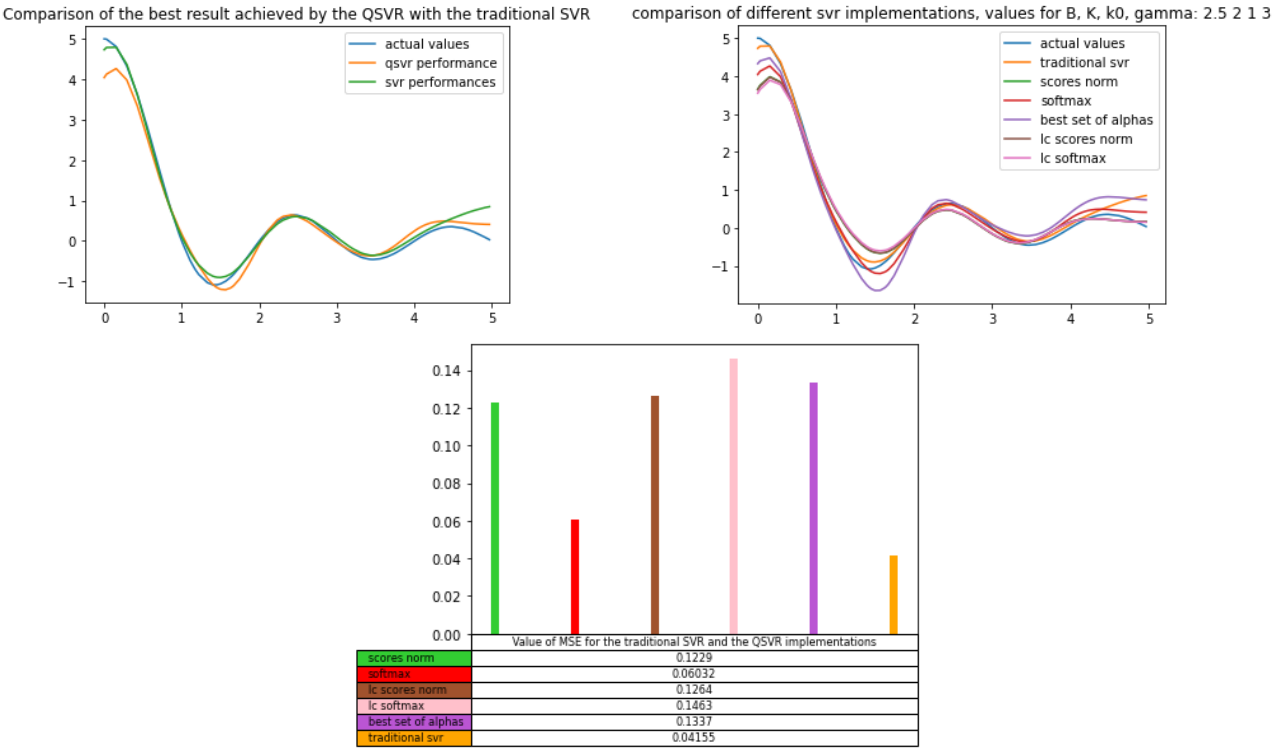


Figure 6.26: Results for test 2 for the function $y = 5\text{sinc}(x)$

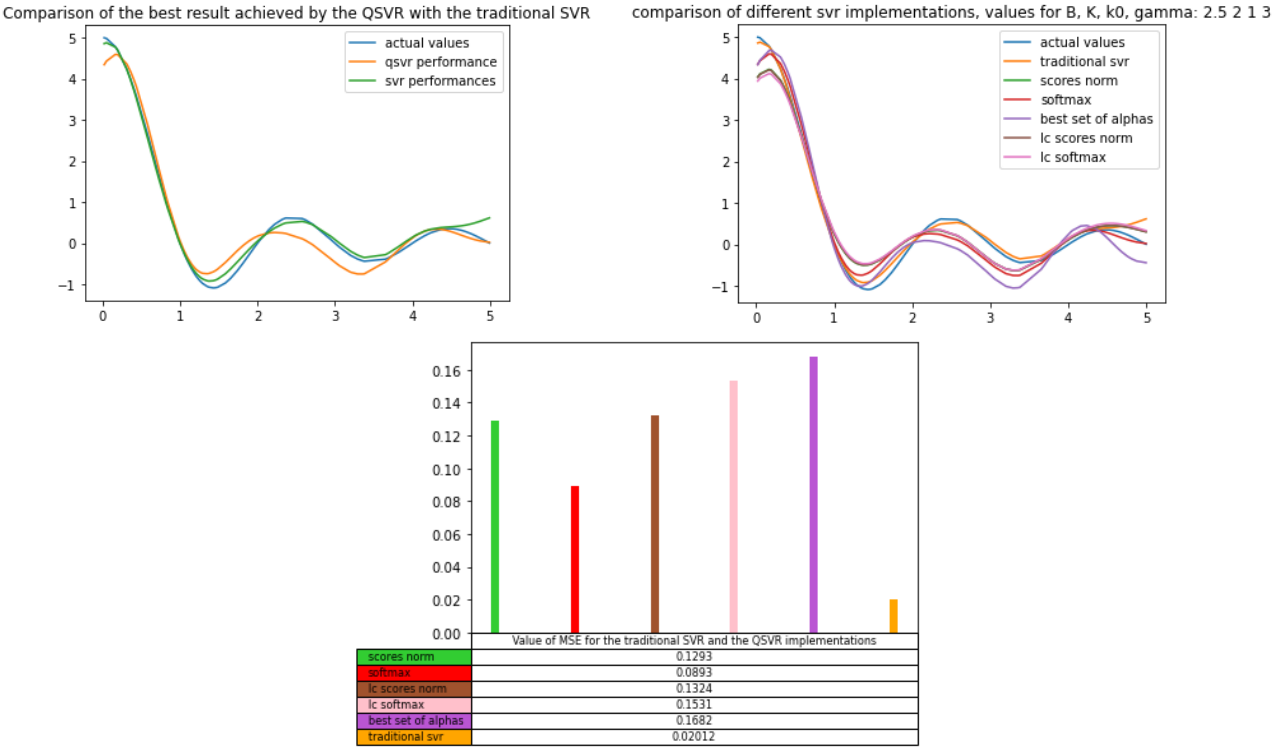


Figure 6.27: Results for test 3 for the function $y = 5\text{sinc}(x)$

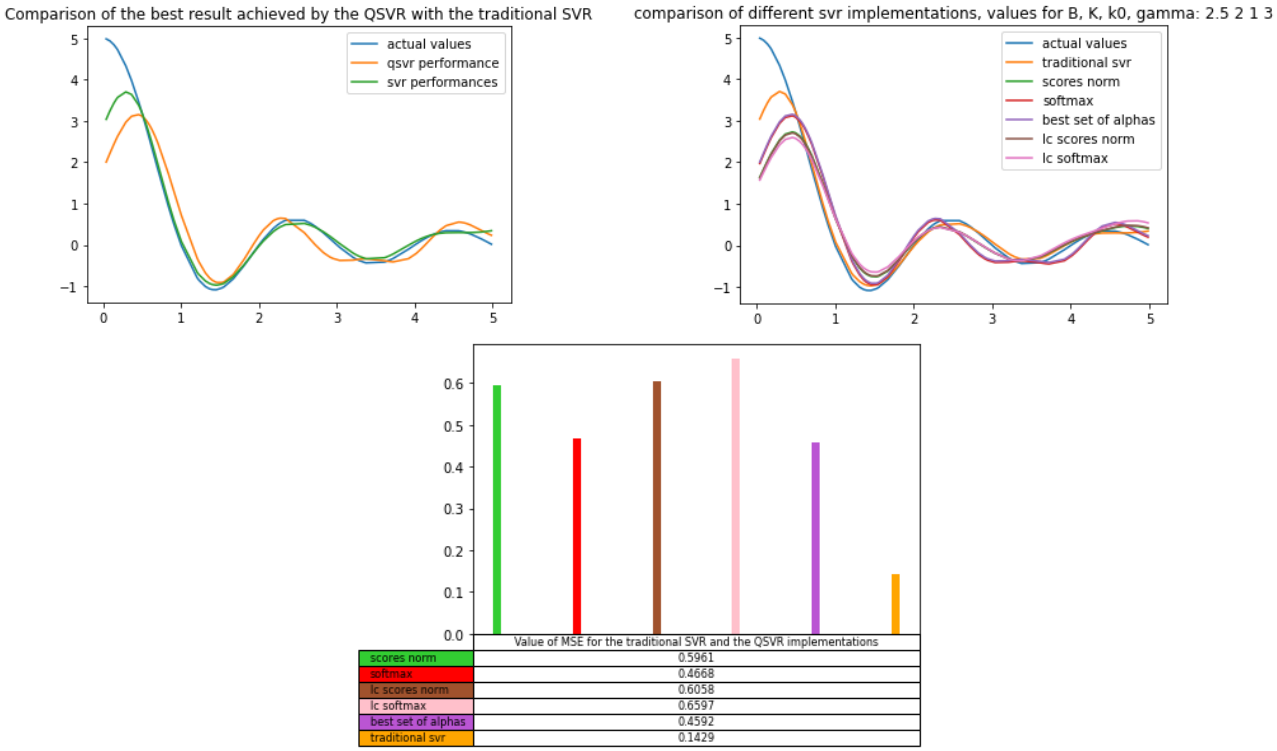


Figure 6.28: Results for test 4 for the function $y = 5\text{sinc}(x)$

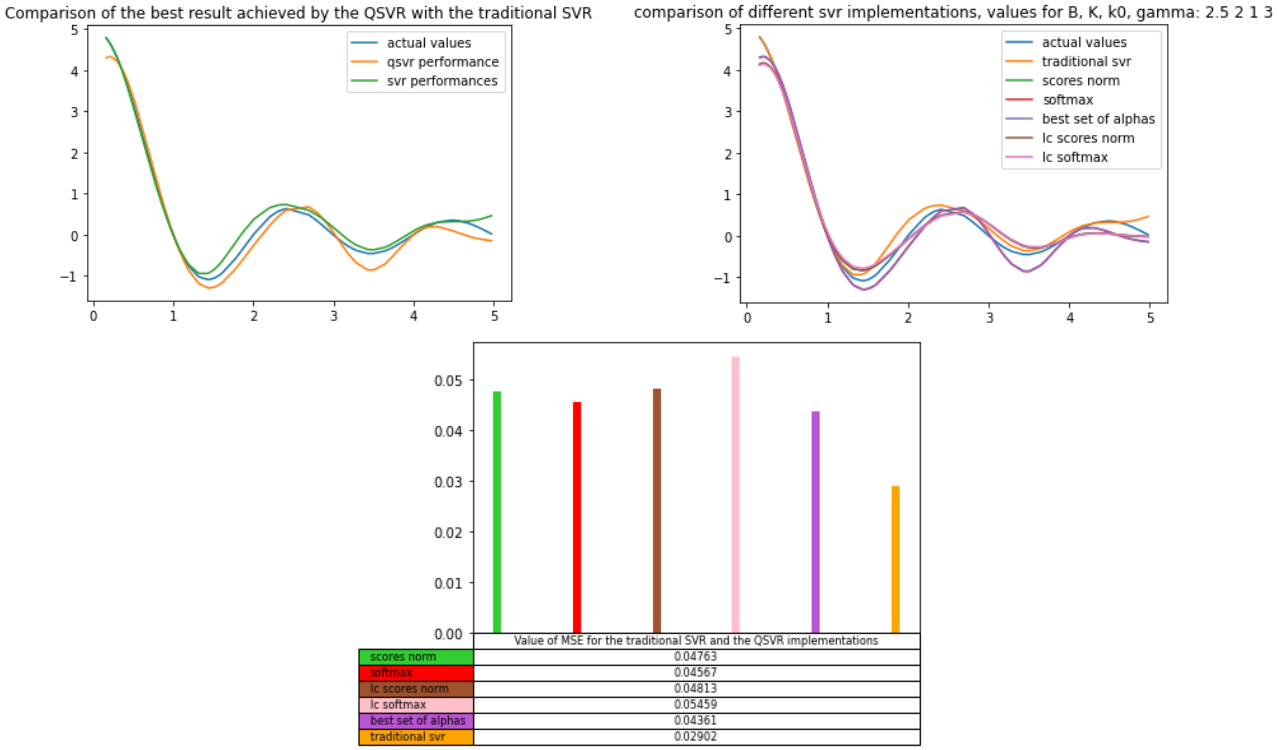


Figure 6.29: Results for test 5 for the function $y = 5\text{sinc}(x)$

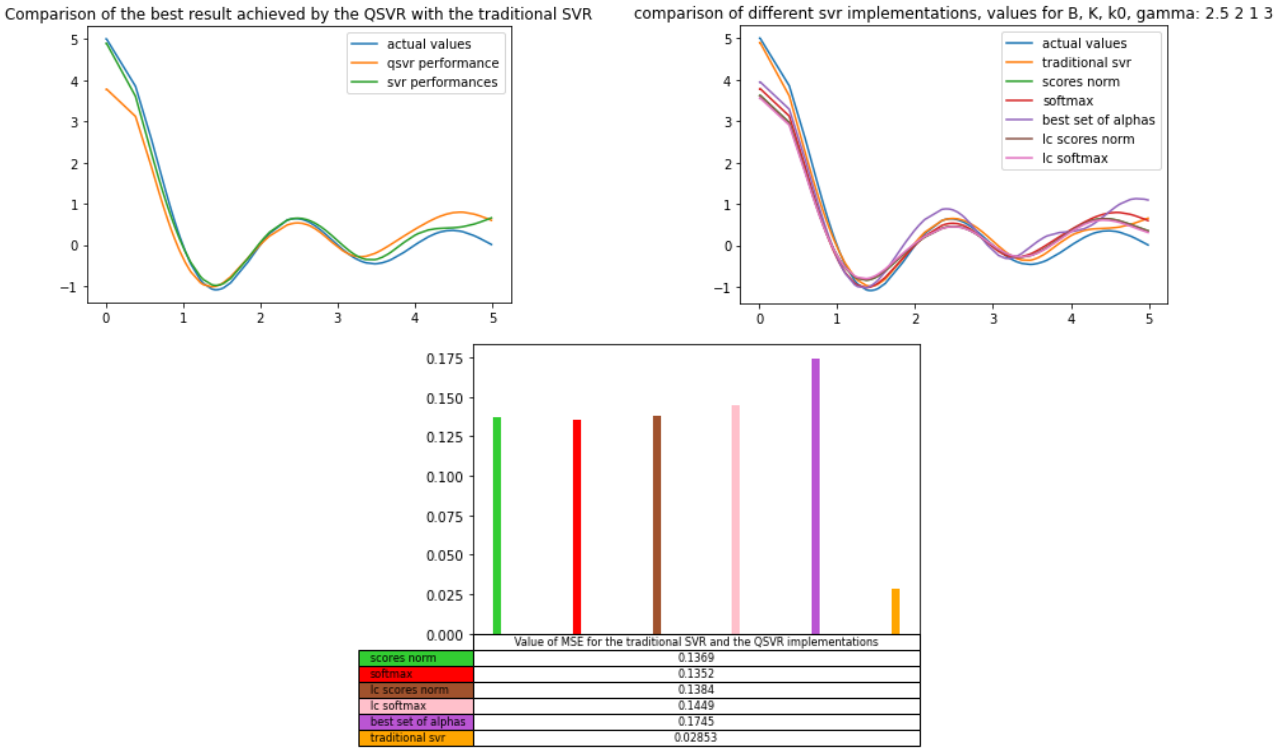


Figure 6.30: Results for test 6 for the function $y = 5\text{sinc}(x)$

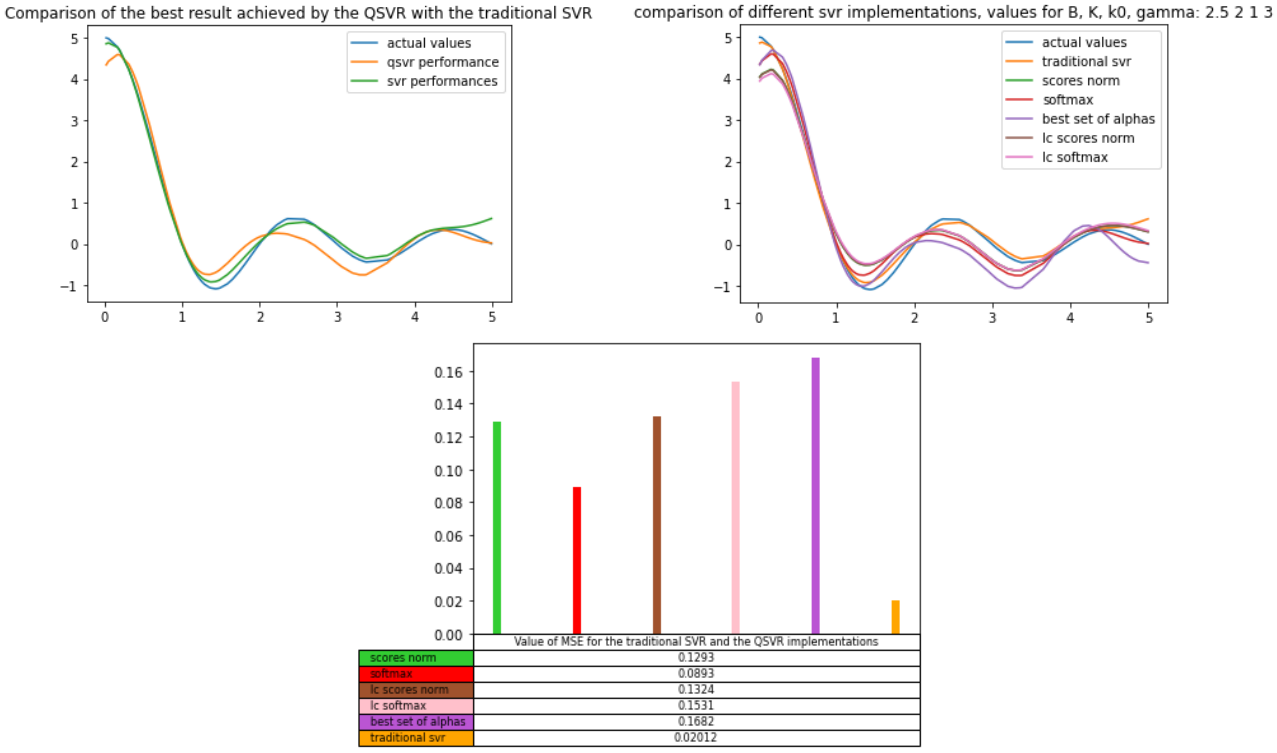


Figure 6.31: Results for test 7 for the function $y = 5\text{sinc}(x)$

By analyzing the numerical results and the depictions of the approximations of the target function it is possible to infer that the performances of the QSVR and SVR highly depend on the training dataset. This dependence on the sampling for the initial dataset is greater with respect to the case of the sine function discussed before. The test cases illustrated in figure 6.26, 6.27 and 6.31 constitute some interesting cases: the approximation of the target function in the region where the slope is greater are better than the previous setting with the uniformly-distanced dataset for the same region. Those

cases, in fact, correspond to a dataset with a higher density of samples in such region. In 6.25 the quantum and classical SVR models failed at approximating the function in the interval $[0, 1]$. This result, however, should not be considered in a bad way: as was pointed out before, in this work was not proposed a new machine learning model but rather a new optimzation method for a pre-existing one, therefore the fact that the QSVR behaved similarly to the SVR should be seen as a good result.

6.3 Test case: triangular function

In this final setting the QSVR is tested for the regression on a triangular function. The formula of such function is given by:

$$\Lambda(x;\alpha) = \begin{cases} \alpha - |x|, & \text{if } |x| < \alpha; \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

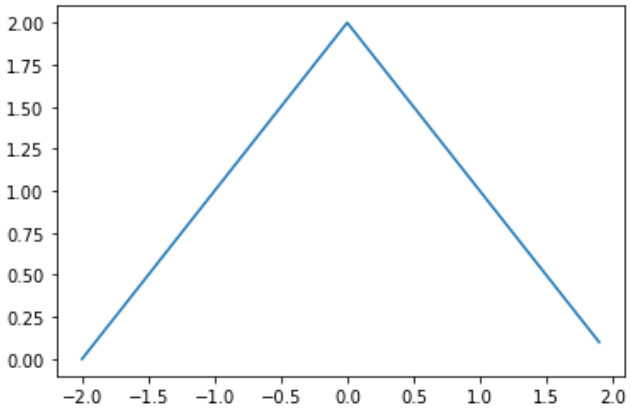


Figure 6.32: Depiction of the considered triangular function

In the proposed test case the function had the form $y = \Lambda(x; 2)$. For this case the values of the hyperparameters were: B=2.0, K=2, $k_0 = 1$ and $\gamma = 1.5$. The value of C is still set to 1000 in order to study the overfitting case.

6.3.1 First test case: evenly spaced training points in the input domain

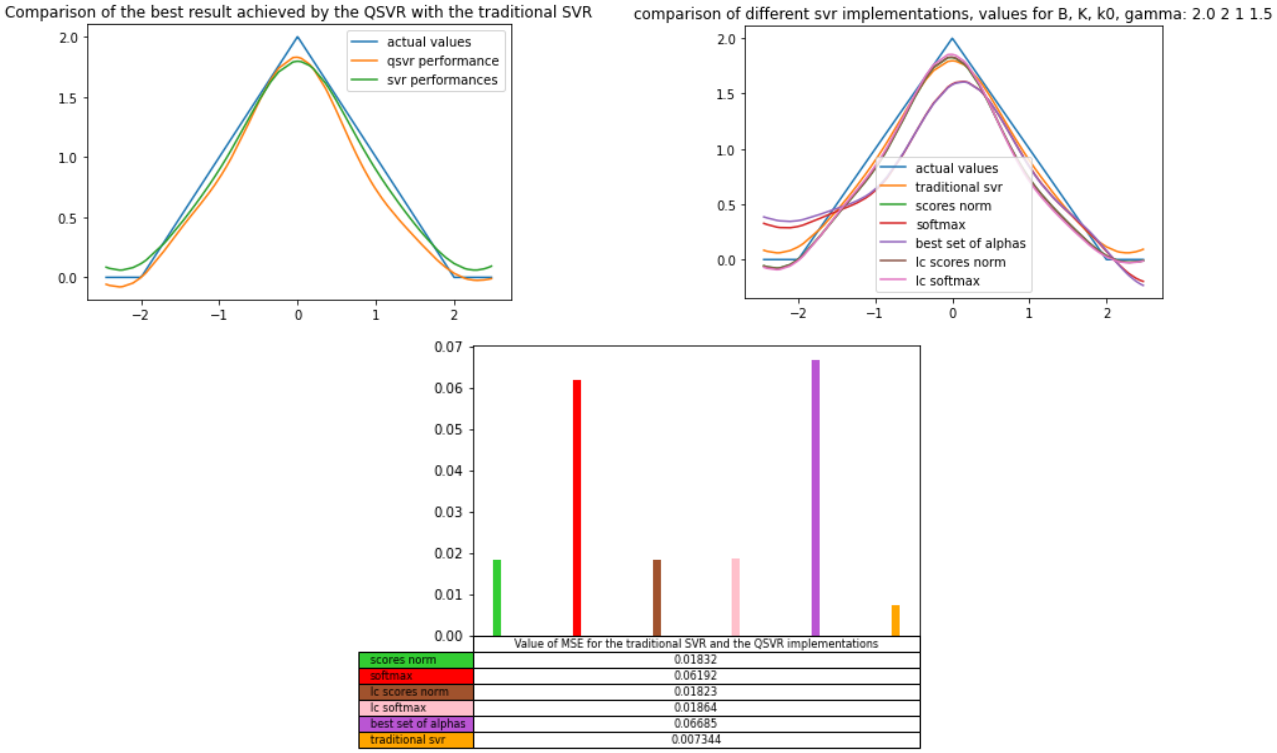


Figure 6.33: Results for test 1 for the function $y = \Lambda(x; 2)$

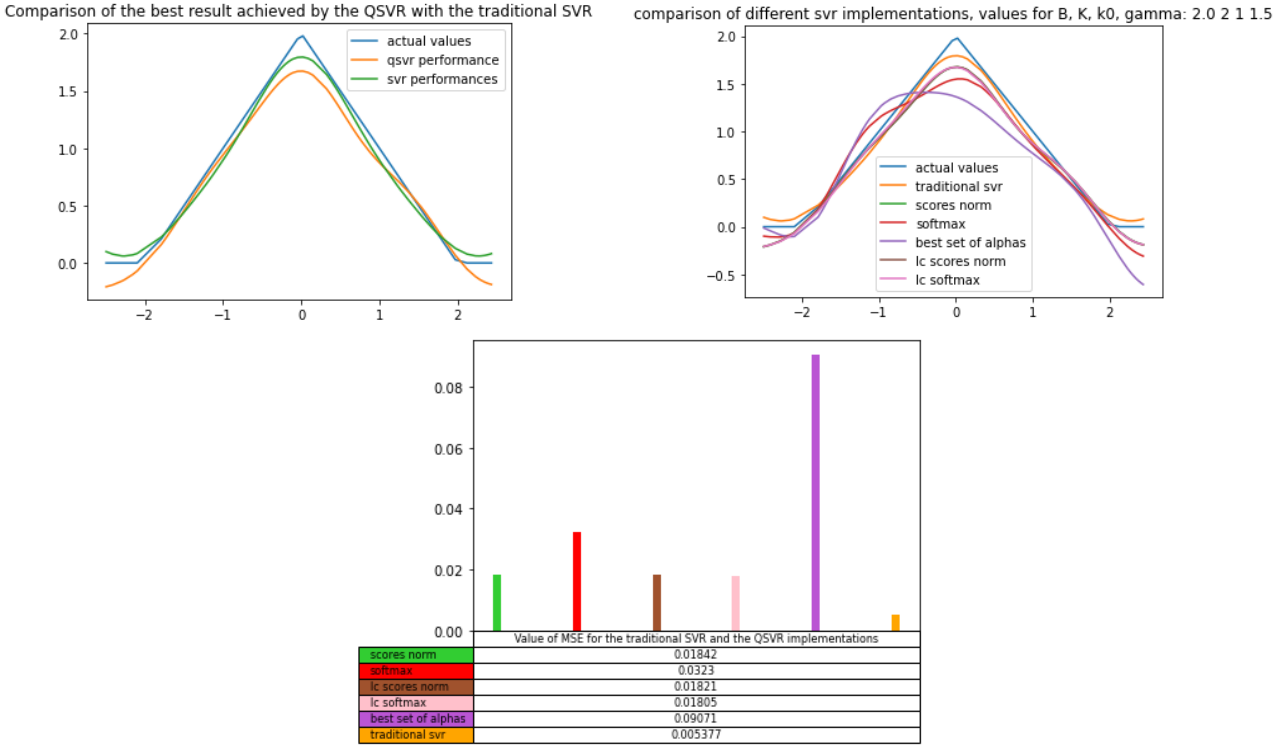


Figure 6.34: Results for test 2 for the function $y = \Lambda(x; 2)$

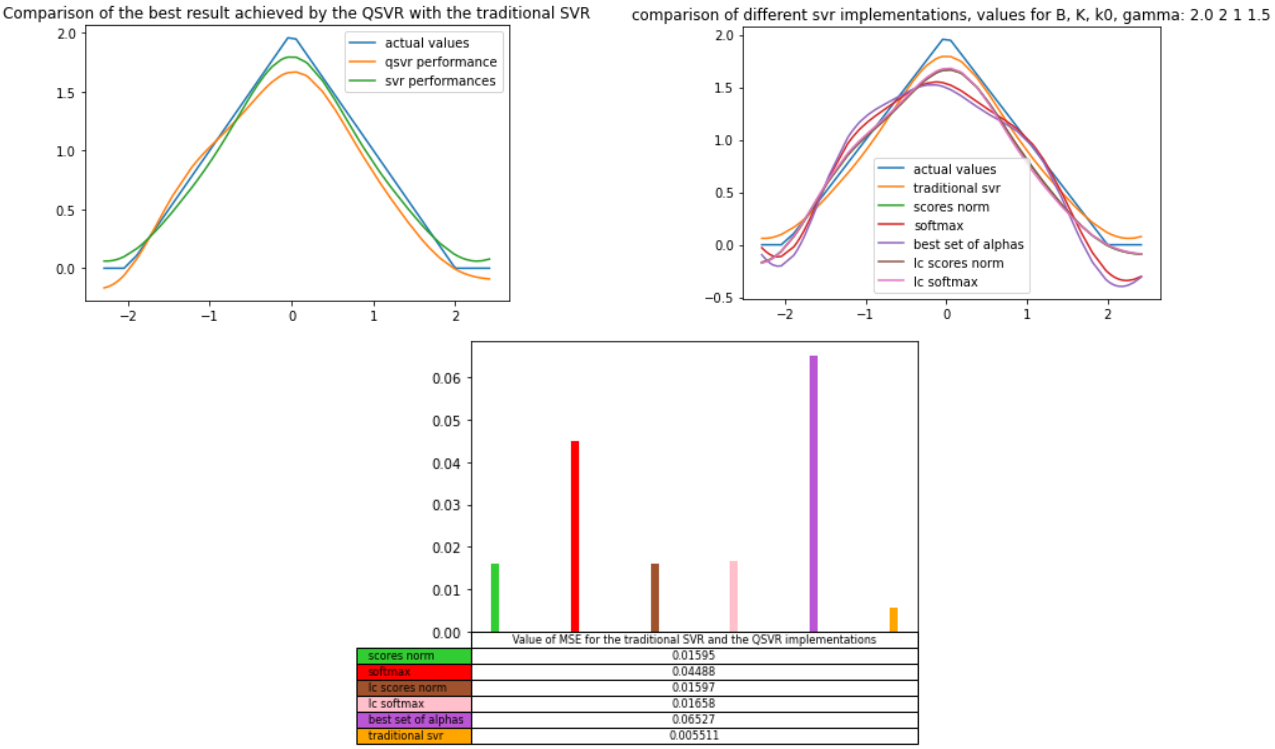


Figure 6.35: Results for test 3 for the function $y = \Lambda(x; 2)$

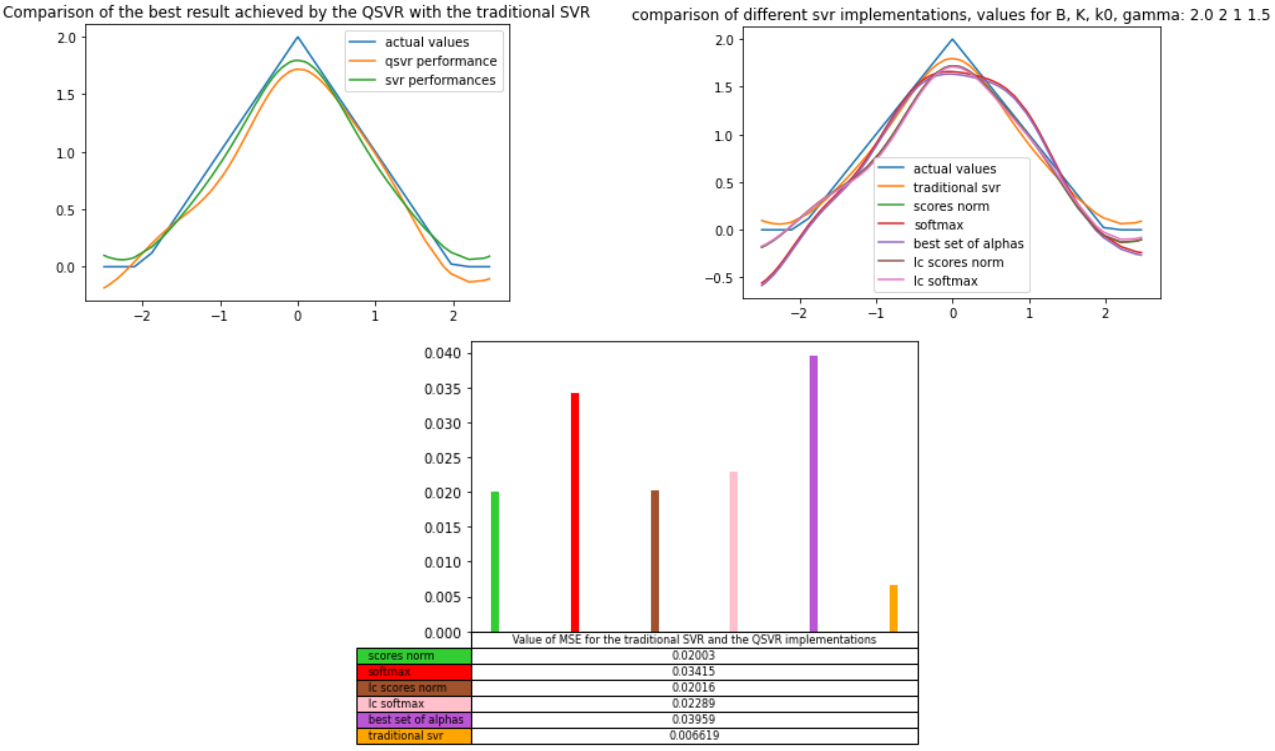


Figure 6.36: Results for test 4 for the function $y = 2\Lambda(x; 2)$

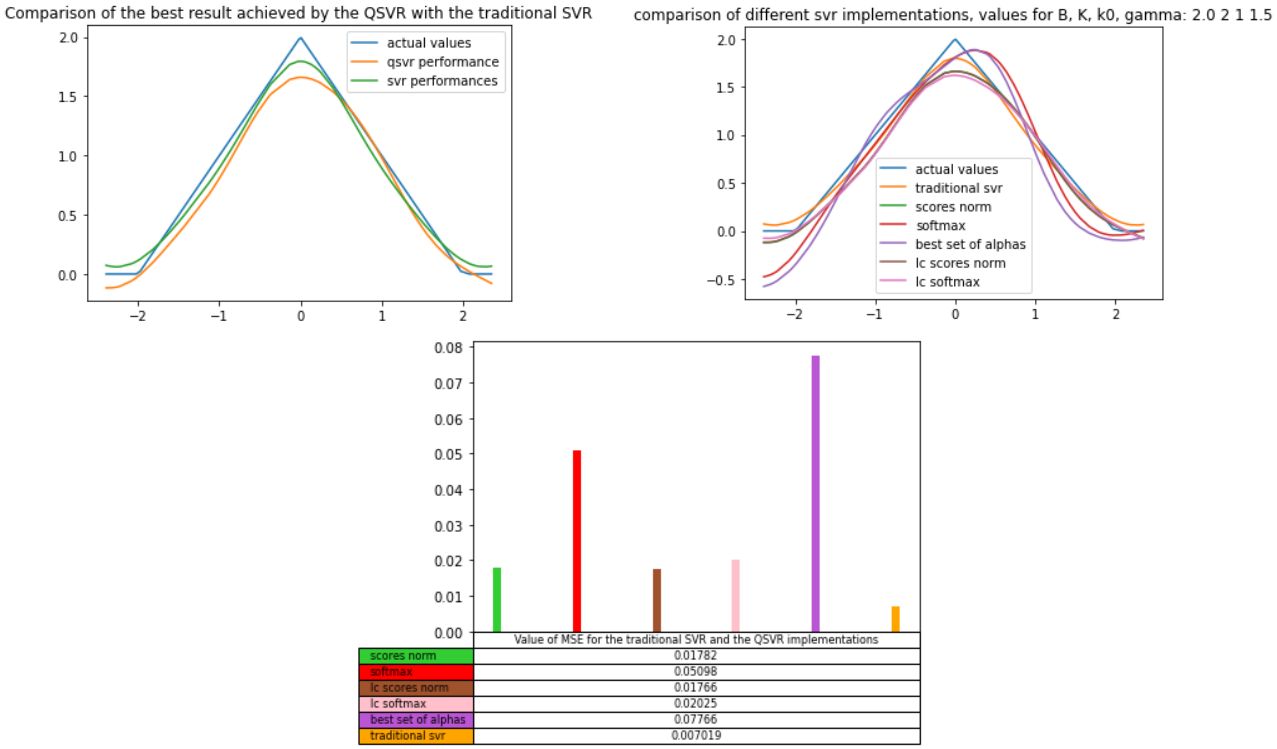


Figure 6.37: Results for test 5 for the function $y = 2\Lambda(x; 2)$

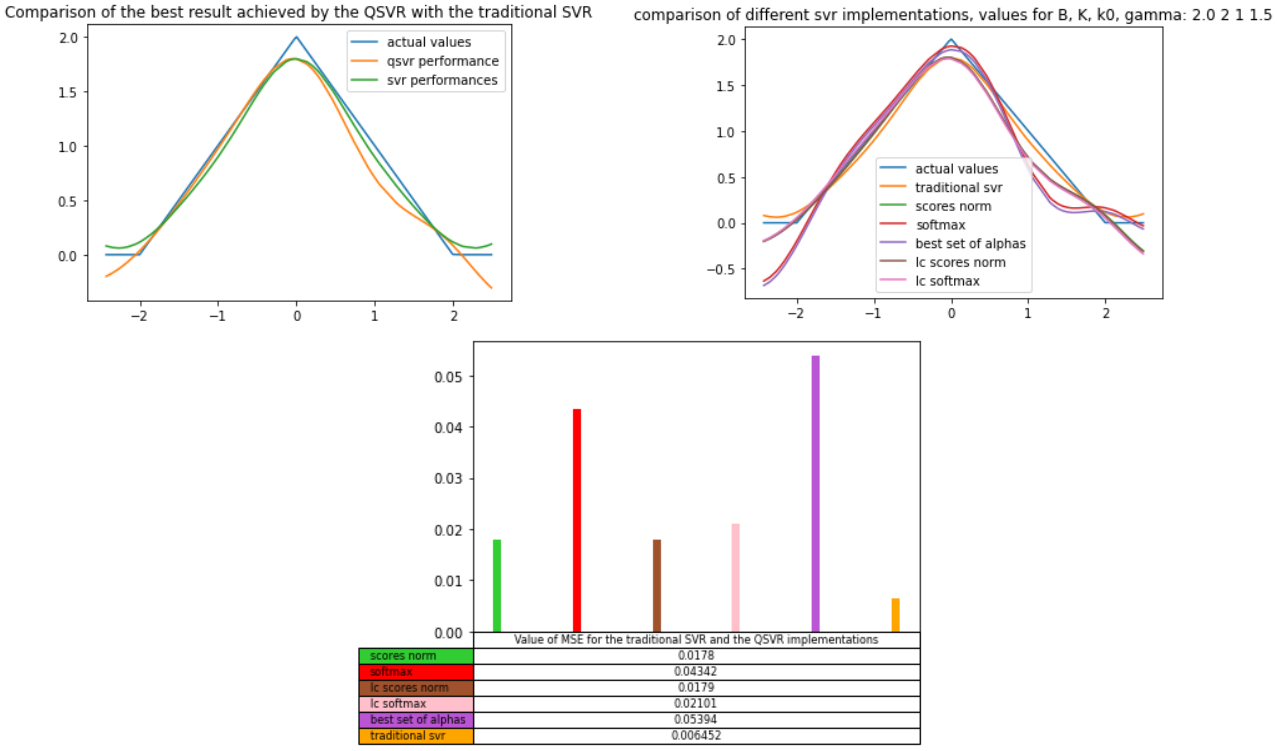


Figure 6.38: Results for test 6 for the function $y = 2\Lambda(x; 2)$

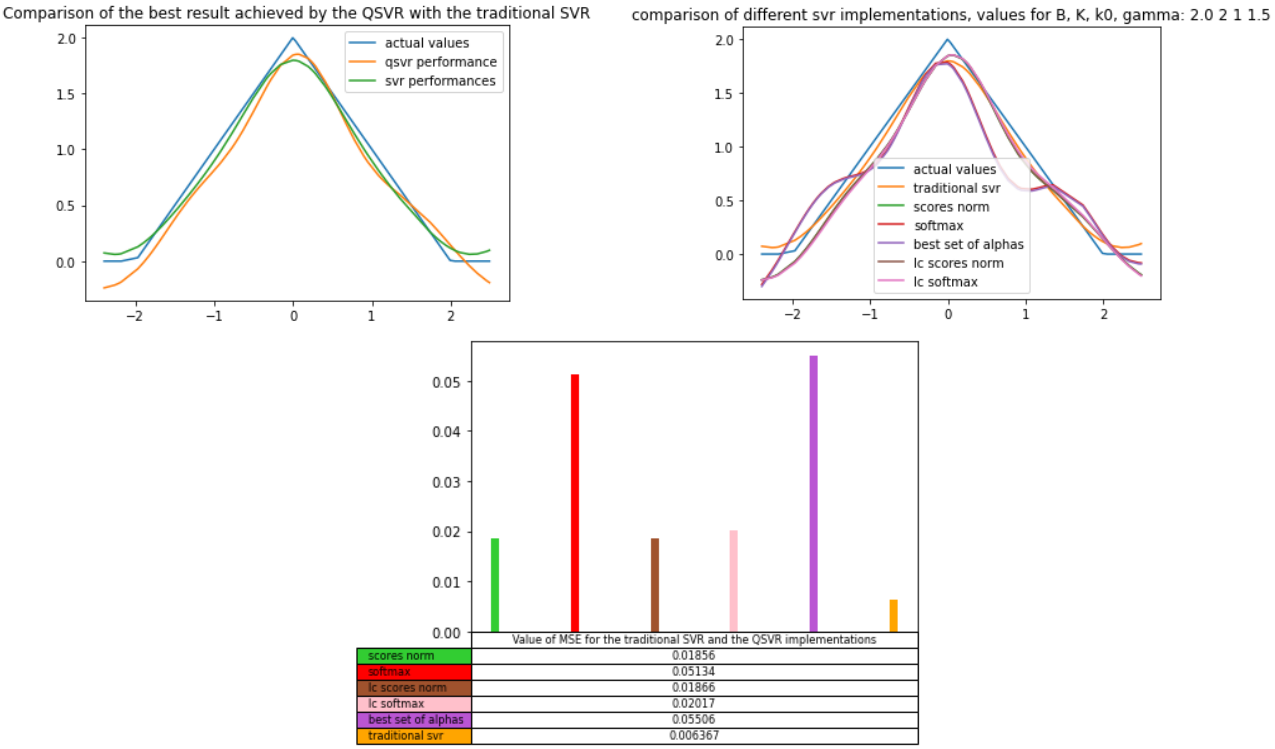


Figure 6.39: Results for test 7 for the function $y = 2\Lambda(x; 2)$

The most evident fact that can be observed from this test case is how bad the solutions combination method *best set of alphas* compared to the other methods. This is an interesting fact when compared to the results described in section 6.1 where the method best set of alphas yielded the best results. This again supports the theory that there is not a best methodology to combine the solutions, but it should be chosen depending on the considered case.

6.3.2 Second test case: randomly sampled dataset

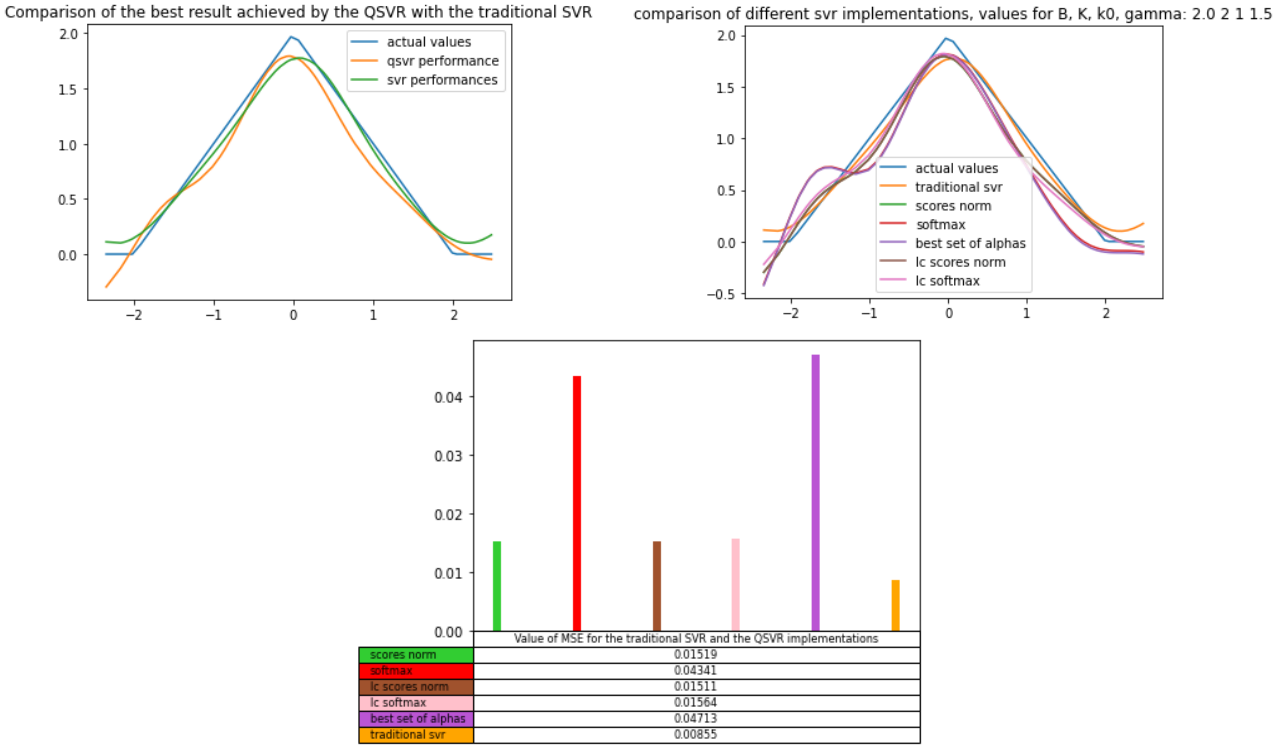


Figure 6.40: Results for test 1 for the function $y = 2\Lambda(x; 2)$

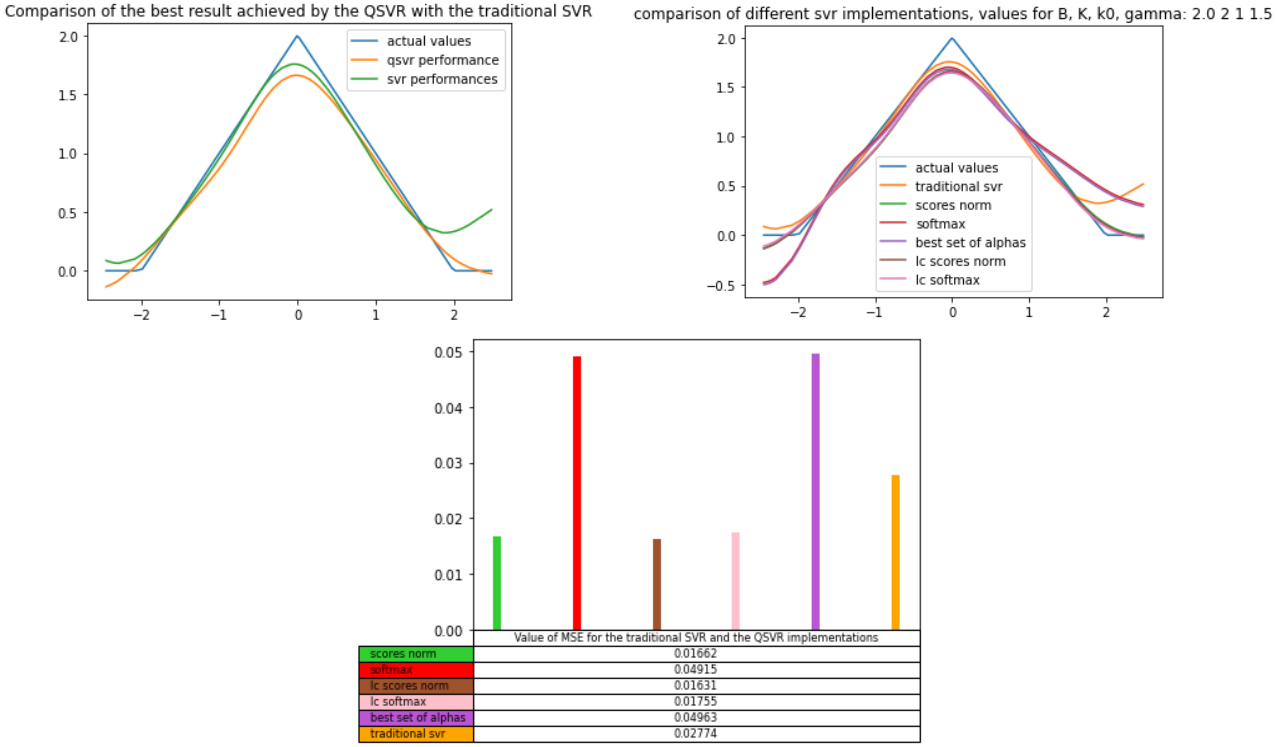


Figure 6.41: Results for test 2 for the function $y = 2\Lambda(x; 2)$

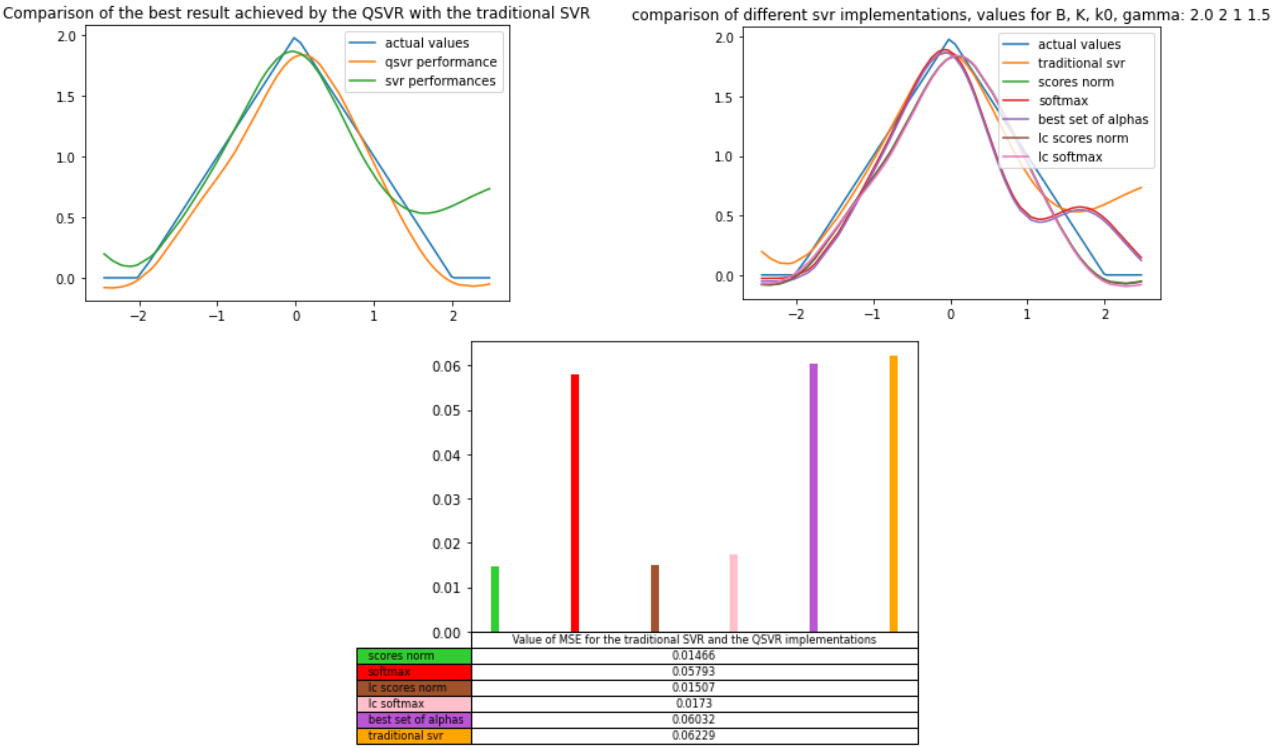


Figure 6.42: Results for test 3 for the function $y = 2\Lambda(x; 2)$

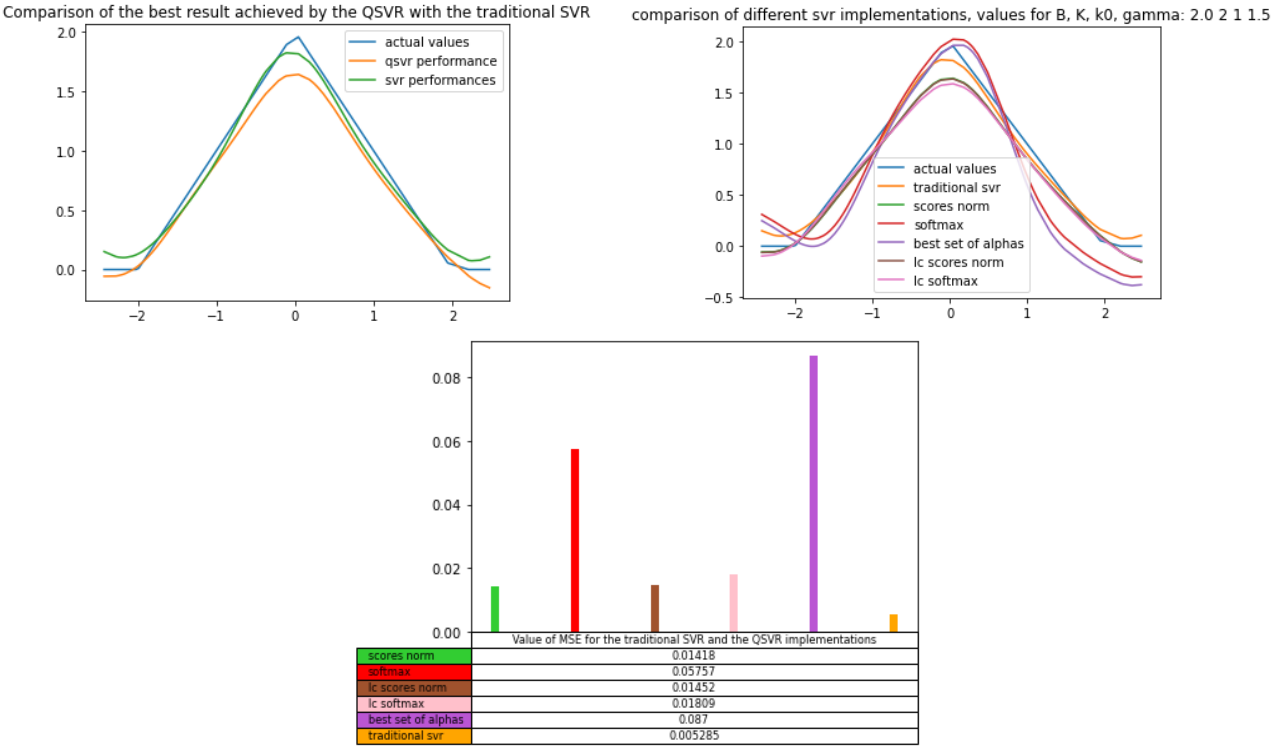


Figure 6.43: Results for test 4 for the function $y = 2\Lambda(x; 2)$

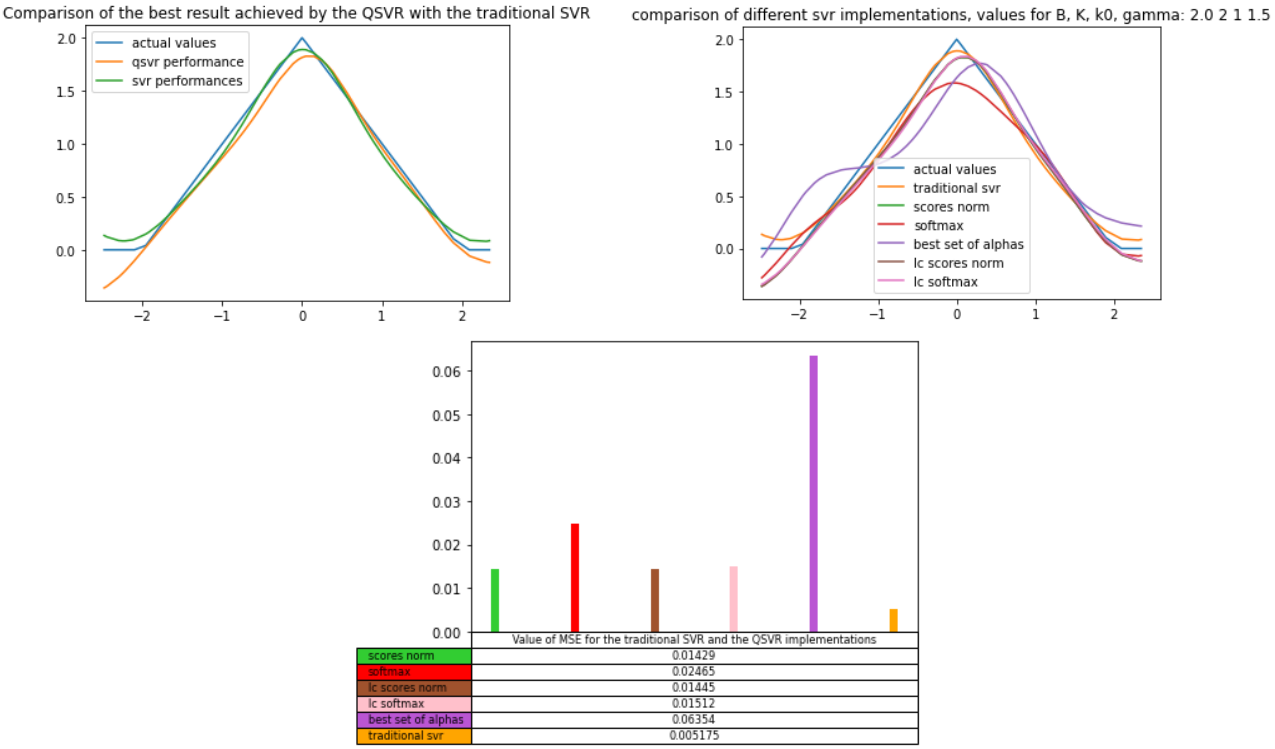


Figure 6.44: Results for test 5 for the function $y = 2\Lambda(x; 2)$

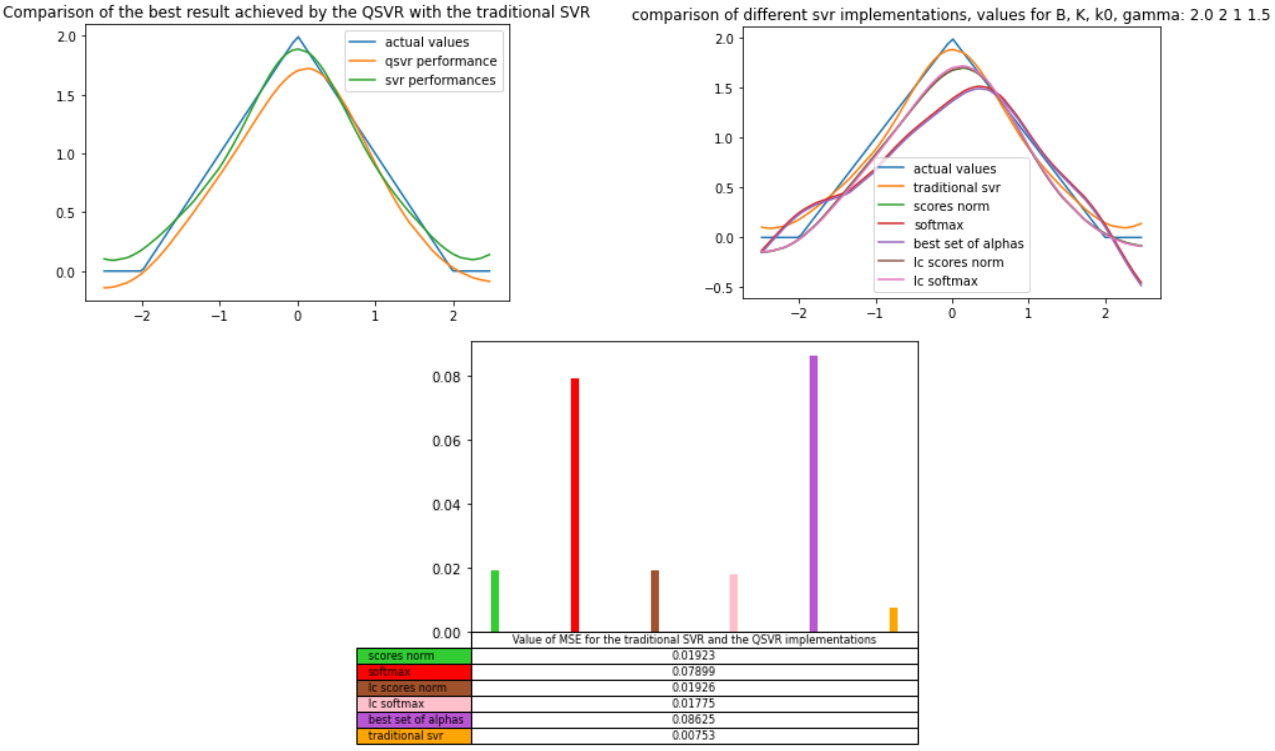


Figure 6.45: Results for test 6 for the function $y = 2\Lambda(x; 2)$

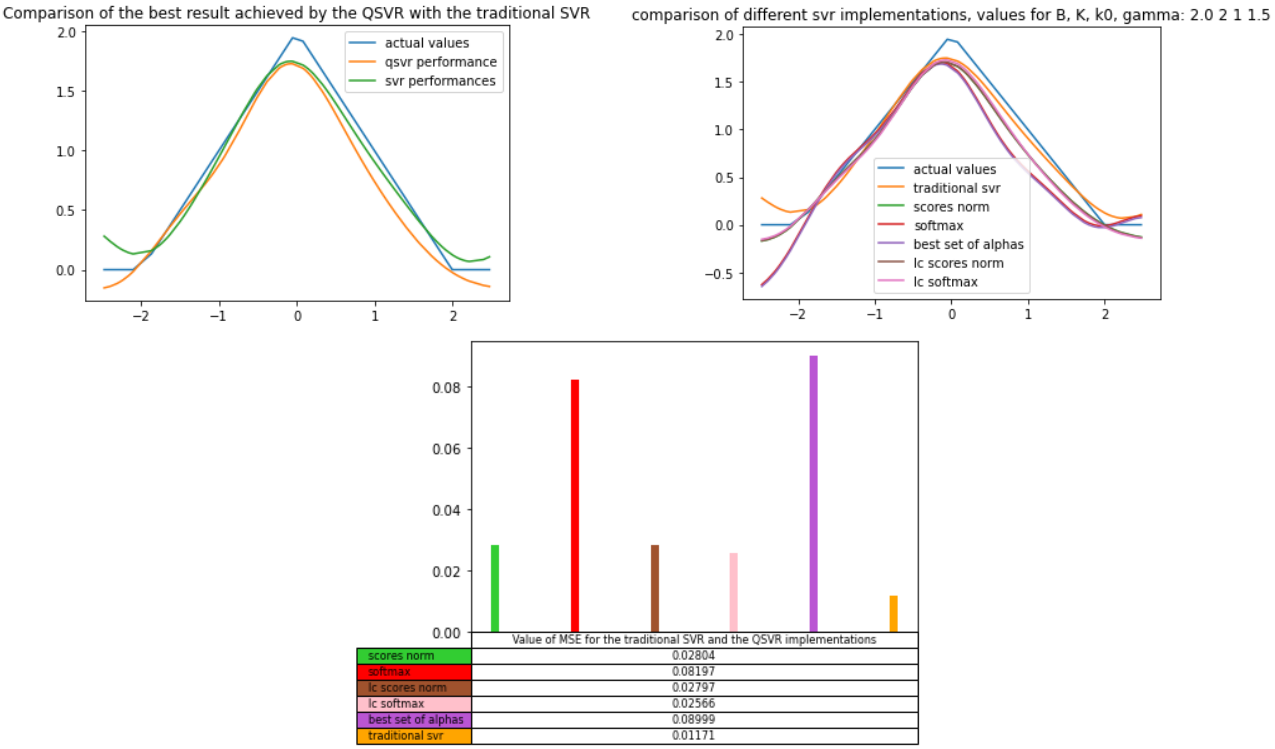


Figure 6.46: Results for test 7 for the function $y = 2\Lambda(x; 2)$

The most relevant fact that can be observed in this test-case is that, for the first time, the implementation of QSVR outperformed the traditional one in the cases illustrated in figure 6.41 and 6.42. For the test run of figure 6.41 the traditional SVR performed better than the softmax and best set of alphas implementation but was surpassed by the others. In the test of figure 6.5 instead, the traditional SVR was surpassed by every implementation of QSVR and the overall best performance was achieved by the methodology scores norm. This event is of great importance for understanting the potentialities of Quantum Annealing as an optimization metaheuristic. However, for fairness, it is also important to point out that the experimental results proposed here constitute only a small part of the experimental validation that was conducted where in most of the cases the traditional SVR still outperformed all the QSVR implementations to some degree. Nevertheless, this remains an encouraging result in the context of the application of QA as an effective methodology for optimization problems in the field of machine learning.

Chapter 7

Conclusions

7.1 Final Considerations

In this work was proposed an implementation of a Support Vector Machine for Regression that employed Quantum Annealing to solve the optimization problem related to the training phase of the algorithm. In order to solve the associated optimization problem it was turned into a QUBO by using an encoding strategy for the variables $\underline{\alpha}$ and $\underline{\hat{\alpha}}$ analogous to the one proposed in [29] for the classification case. The main issue in this configuration is that the values that the $\underline{\alpha}$ and $\underline{\hat{\alpha}}$ can take are discretized and limited in number according the formulae 5.4 and 5.5. In [29] was noted that this fact might not be of great importance in the classification because "the global order of magnitude of all α_n is often not as important as the relative factors between different α_n". In the regression case, however, this might not be the case because the object of interest is for the estimation of the target variables to be the most precise as possible and therefore the values of the α_n and $\hat{\alpha}_n$ themselves might be of more importance rather than their relative factors. In [29] was noted that the usage of a negative exponent of the basis for the encoding, i.e. 5.4 and 5.5, was not necessary for the achievement of good results, however, in the experimental analysis that was conducted for this work the usage of the parameter k_0 in the encoding provided better results if chosen correctly. This observation might support the theory that in the regression case the exact value of the coefficients $\underline{\alpha}$ and $\underline{\hat{\alpha}}$ have a more important role tahn they do in the classification case. The various implementations of the Quantum SVR also proved to be more sensible to the tuning of the hyperparameters, especially for γ, that controlled the shape of the kernel function. This characteristic of the QSVR could also be ascribed to the discretization of the α_n and $\hat{\alpha}_n$, they in fact, can only take a limited set of values and therefore might not adapt well to different values of γ. Another fact worthy of note concerns the choice of the value of K, i.e. the number of logical qubits used for encoding each variable of the original problem. It is expected, in fact, that the larger the value of K, and therefore the larger the number of values that the α_n and $\hat{\alpha}_n$ can take, the better the precision in the regression. However, empirical validation showed that choosing too large a value for K resulted in poor performances in the prediction of the test set. However, it should be noted that the experimental validation was conducted on simple functions of one variable with a small number of training samples. A deeper study on the effect of hyperparameters tuning should be performed in order to gain a better understanding of their effect on the performances of the regression process. In section 6.3 the QSVR was applied to the estimation of a triangular function and in some instances it managed to even surpass the the traditional implementation of SVR(figure 6.41 and 6.42). Even though in the majority of the experimental runs the classical SVR still performed slightly better or similarly to the quantum counterpart this achievement is of of great importance to understand the capabilities of Quantum Annealing in the context of optimization problems. It worth noting, in fact, that since the original problem was continous and constrained the problem itself had to undergo an important transformation in order to be solved by the annealer. Therefore, the fact that the annealer managed to achieve results that are similar or in some cases even better than the classical counterpart even though the optimization problem is very different from its original formulation is remarkable and indicates the potentialities of QA.

7.2 Further Developments

The test cases analyzed in this thesis are very simple problems and are of little interest in the context of practical applications. Nevertheless, this analysis constitutes an interesting starting point for the study of the potentialities of QA-based SVR. Future research on this field will investigate the application of it to more complex and bigger dataset and will assess the result of applying such methodologies to problem instances of practical interest. The development of a reliable and scalable implementation of a QSVR would be of great importance in the context of regression analysis and ML. In the future the study of more complex problems will also require a larger number of qubits and couplers for the problem formulation and therefore an effective application of the proposed QSVR implementation in these contexts highly depends upon the availability of quantum hardware powerful enough to sustain such a complexity in terms of computation.

7.3 Conclusions

In conclusion, QA established itself as a powerful tool for optimization problems and a valuable asset for machine learning applications. The main drawback, epsecially when dealing with ML applications, consists of the fact that considered problem must be in the QUBO form. Moreover, QA, as was pointed out in [14], is a tool mainly used to optimize traditional machine learning algorithm in the training phase rather than building new ones. In this work the encoding strategy was inspired from an implementation that was originally designed for classification applications, therefore in the future other encoding strategies could be investigated in order to improve the expressive power of the regression algorithm. Further research will also investigate the application of QSVR on bigger and more complex problems while also trying to make some improvements on the problem formulation.

Chapter 8

Acknowledgements

The thesis work was developed in collaboration with the Jülich Supercomputing center (JSC). I gratefully acknowledge the Jülich Supercomputing Centre for funding this project by providing computing time on the D-Wave Advantage system through the Jülich UNified Infrastructure for Quantum computing (JUNIQ)

Bibliography

[1] Steven H. Adachi and Maxwell P. Henderson. *Application of Quantum Annealing to Training of Deep Neural Networks*. 2015. arXiv: 1510.06356 [quant-ph].

[2] B. Apolloni, C. Carvalho, and D. de Falco. "Quantum stochastic optimization". In: *Stochastic Processes and their Applications* 33.2 (Dec. 1989), pp. 233–244. DOI: 10.1016/0304-4149(89)90040-9. URL: https://doi.org/10.1016/0304-4149(89)90040-9.

[3] Leonora Bianchi et al. "A survey on metaheuristics for stochastic combinatorial optimization". In: *Natural Computing* 8.2 (Sept. 2008), pp. 239–287. DOI: 10.1007/s11047-008-9098-4. URL: https://doi.org/10.1007/s11047-008-9098-4.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] M. Born and V. Fock. "Beweis des Adiabatensatzes". In: *Zeitschrift fr Physik* 51.3-4 (Mar. 1928), pp. 165–180. DOI: 10.1007/bf01343193. URL: https://doi.org/10.1007/bf01343193.

[6] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, 1992, pp. 144–152.

[7] Gabriele Cavallaro et al. "Approaching Remote Sensing Image Classification with Ensembles of Support Vector Machines on the D-Wave Quantum Annealer". In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. 2020, pp. 1973–1976. DOI: 10.1109/IGARSS39084.2020.9323544.

[8] Prasanna Date and Thomas E. Potok. "Adiabatic Quantum Linear Regression". In: *CoRR* abs/2008.02355 (2020). arXiv: 2008.02355. URL: https://arxiv.org/abs/2008.02355.

[9] H. Drucker et al. "Support Vector Regression Machines". In: *NIPS*. 1996.

[10] Diego de Falco, B. Apolloni, and Nicolò Cesa-Bianchi. "A numerical implementation of quantum annealing". In: July 1988.

[11] E. Farhi et al. "Quantum Computation by Adiabatic Evolution". In: *arXiv: Quantum Physics* (2000).

[12] Robert C. Foster, Brian Weaver, and James Gattiker. *Applications of Quantum Annealing in Statistics*. 2019. arXiv: 1904.06819 [stat.CO].

[13] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[14] Wen Guan et al. "Quantum machine learning in high energy physics". In: *Machine Learning: Science and Technology* 2.1 (Mar. 2021), p. 011003. DOI: 10.1088/2632-2153/abc17d. URL: https://doi.org/10.1088/2632-2153/abc17d.

[15] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (Oct. 2009). ISSN: 1079-7114. DOI: 10.1103/physrevlett.103.150502. URL: http://dx.doi.org/10.1103/PhysRevLett.103.150502.

[16] Helmut G. Katzgraber et al. "Seeking Quantum Speedup Through Spin Glasses: The Good, the Bad, and the Ugly". In: *Physical Review X* 5.3 (Sept. 2015). ISSN: 2160-3308. DOI: 10.1103/physrevx.5.031026. URL: http://dx.doi.org/10.1103/PhysRevX.5.031026.

[17] Richard Y. Li et al. "Quantum annealing versus classical machine learning applied to a simplified computational biology problem". In: *npj Quantum Information* 4.1 (Feb. 2018). DOI: 10.1038/s41534-018-0060-8. URL: https://doi.org/10.1038/s41534-018-0060-8.

[18] Catherine McGeoch and Pau Farre. *The D-Wave Advantage System: An Overview*. Tech. rep. 2020.

[19] Tom Mitchell. *Machine Learning*. McGraw-Hill Education, 1997.

[20] Hartmut Neven et al. *Training a Binary Classifier with the Quantum Adiabatic Algorithm*. 2008. arXiv: 0811.0416 [quant-ph].

[21] Ziad Obermeyer and Ezekiel J. Emanuel. "Predicting the Future — Big Data, Machine Learning, and Clinical Medicine". In: *New England Journal of Medicine* 375.13 (Sept. 2016), pp. 1216–1219. DOI: 10.1056/nejmp1606181. URL: https://doi.org/10.1056/nejmp1606181.

[22] "Quantum theory, the Church–Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (July 1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070. URL: https://doi.org/10.1098/rspa.1985.0070.

[23] "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (Dec. 1992), pp. 553–558. DOI: 10.1098/rspa.1992.0167. URL: https://doi.org/10.1098/rspa.1992.0167.

[24] A. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM J. Res. Dev.* 3 (1959), pp. 210–229.

[25] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "Prediction by linear regression on a quantum computer". In: *Physical Review A* 94.2 (Aug. 2016). ISSN: 2469-9934. DOI: 10.1103/physreva.94.022342. URL: http://dx.doi.org/10.1103/PhysRevA.94.022342.

[26] Fabio Sebastiano et al. "Cryo-CMOS Electronic Control for Scalable Quantum Computing: Invited". In: June 2017, pp. 1–6. DOI: 10.1145/3061639.3072948.

[27] Abhinav Sharma et al. "Machine Learning Applications for Precision Agriculture: A Comprehensive Review". In: *IEEE Access* 9 (2021), pp. 4843–4873. DOI: 10.1109/ACCESS.2020.3048415.

[28] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172.

[29] D. Willsch et al. "Support vector machines on the D-Wave quantum annealer". In: *Computer Physics Communications* 248 (Mar. 2020), p. 107006. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2019.107006. URL: http://dx.doi.org/10.1016/j.cpc.2019.107006.

[30] Dennis Willsch et al. *Benchmarking Advantage and D-Wave 2000Q quantum annealers with exact cover problems*. 2021. arXiv: 2105.02208 [quant-ph].

[31] Adonis Yatchew. "Nonparametric Regression Techniques in Economics". In: *Journal of Economic Literature* 36.2 (June 1998), pp. 669–721. URL: https://ideas.repec.org/a/aea/jeclit/v36y1998i2p669-721.html.