UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

INSTITUTE OF MATHEMATICAL STATISTICS

Vassili Mušnikov

ON THE USAGE OF SUPPORT VECTOR MACHINES FOR THE SHORT-TERM PRICE MOVEMENT PREDICTION IN INTRA-DAY TRADING

Master's thesis

Supervisor: Raul Kangro, PhD

TARTU 2013

Contents

Introduction

In our changing modern society grows the significance of financial issues. Entities and individuals speculate on the stock exchanges in order to trade financial securities and to gain profit. We all know the words of a great politician Benjamin Franklin who once said: “Time is money.” The fast developing world of information technologies and computers gives an opportunity to do many things in a short period of time. Modern financial markets represent electronic networks which give an advantage to perform transactions at increased speed and reduced cost. When a company decides to buy or sell the large amount of shares, this may lead to the unwanted superfluous loss. In order to minimize the loss, it is needed to develop a trading algorithm which can generate buy or sell signals using future stock price forecasts. That is the reason why the prediction of stock price is necessary in trade executions. That brings me to the aim of the underlying thesis.

The aim of the current thesis is to research the prediction of future stock prices by using the implementation of support vector machines, also to find possible technical solutions and to interpret the gained results. In order to consider the problem of forecasting future stock prices for a short period of time, the market data of the British multinational telecommunications company Vodafone Group Plc and the British-Swedish multinational pharmaceutical and biologics company AstraZeneca Plc is being used to fit the models and verify how good their predictive power is. The opportunities of packages *e1071* and *kernlab* of programming language R are being used in the current thesis. The implementation of the predictions to trading algorithms is not being considered due to it is not relevant to the underlying thesis.

The thesis consists of three chapters. The first chapter is dedicated to support vector machines, because this particular method is used in developing prediction algorithms. For better understanding of the principle of this method, certain fundamentals are being explained. The first chapter introduces what is machine learning, explains finding the regression function by using support vector machines and mentions the problems which may arise during finding the regression function. The concept of regression estimation is being explained with theoretical and graphical examples.

The second chapter is dedicated to kernels, because that gives an opportunity to use non-linear functions as regression functions. In this chapter, the classification of kernels is being introduced. In addition, it is explained to the reader why does the usage of kernel functions simplify the finding of the regression function. The short overview of technical opportunities

of programming language R packages is also being introduced in the second chapter. Finally, such statistical method of evaluating and comparing learning algorithms as cross-validation is being briefly mentioned in the chapter.

Unlike from the first two chapters, which give a theoretical overview, the third chapter is the practical part of the thesis. It introduces the implementation of support vector machines on the short-term price movement prediction in intra-day trading. The algorithm of the price prediction is being explained in the third chapter. Given data is also described in this chapter. Due to similar data involved, the author also presents the comparison with the master's thesis of Andrei Orlov [1].

In addition, at the end of the thesis, the reader can find Appendices which consist of data frame, the diagram explaining the relations between functions in a code of algorithm, the codes of figures and the CD containing the code of the algorithm.

1 Support Vector Machines

1.1 Machine Learning

There is no well accepted definition of what is and what is not *Machine Learning*. But here is a definition of what is Machine Learning according to Webster: "Machine Learning is the ability of machine or a system to improve its performance based on previous results". From mathematical point of view the definition is not quite correct, but it gives a basic overview about the subject.

There are several different types of learning algorithms, but the main two types that Machine Learning can be subdivided into are *supervised learning* and *unsupervised learning*. The term supervised learning refers to the fact that the algorithm is provided with the data set in which the "right answers" are given. Concretely, for each observation a desired output value is given. Learning algorithm analyzes this data set and produces a model, which can be used for mapping yet unseen data. In unsupervised learning the algorithm is trained on a data with no output values. The learning algorithm tries to find hidden structure in unlabeled data [2].

Further, supervised learning can be divided into *classification* and *regression problem*. Classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a data set containing observations whose category is known [3]. In regression problem it is tried to predict a continuous variable instead of a category. The underlying thesis describes an algorithm of a supervised learning producing a regression analysis of a data.

1.2 Loss Functions

In order to produce a regression analysis one usually divides a whole data set into two subsets: *training set*, on which the predictive model is applied, and *test set*, on which the found model is being tested. Often the training data is given as follows: $\{(x_1, y_1), \dots, (x_m, y_m)\}$, where x_i denotes a vector of features and y_i denotes an additional feature for which the predictions will be made. In regression analysis x_i is called a vector of independent variables and y_i is called dependent variable. A regression function $f(x)$ is constructed which is technically a prediction function.

In order to estimate a regression function $f(x)$ which is a function of the independent variables, one usually tries to minimize the empirical risk:

$$R_{emp}(f) = \frac{1}{m}\sum_{i=1}^{m} L(f(x), y), \tag{1}$$

where $L(f(x), y)$ is so-called loss function [4].

In order to give the basic intuition, a regression function $f(x)$ in a class of linear functions is being described by

$$f(x) = \langle w, x \rangle + b, \tag{2}$$

where w is a vector, b is a real number and $\langle w, x \rangle$ represents a dot product between two vectors.

One possible approach for such class is a linear regression model. In case of this model it is tried to fit a linear function to an observed data, using least square method. The aim of this method is to minimize the sum of the squares of the errors made by fitting a linear function. Thus, to get a proper estimation for a regression function $f(x)$ in the case of linear regression model one needs to minimize the following empirical risk equation:

$$R_{emp}(f) = \frac{1}{m}\sum_{i=1}^{m} (f(x_i) - y_i)^2. \tag{3}$$

However, there is a different approach, which shall be used in the underlying thesis. Concretely, Vapnik's ε-insensitive loss function:

$$L(f(x), y) = |y - f(x)|_\varepsilon = max\{0, |y - f(x)| - \varepsilon\}. \tag{4}$$

In other words, the loss is equal to zero if the difference between an observed feature y and the predictive function $f(x)$ is less or equal to ε [5, p. 251].

1.3 Quadratic optimization problem

If ε is sufficiently large, then in the case of (4) empirical risk (1) can be zero. But this doesn't imply that the regression function $f(x)$ is a good predictor. To overcome this problem one

may estimate the regression function $f(x)$ by finding a solution to the following quadratic optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{subject to } \begin{cases} f(x_i) - y_i \leq \varepsilon \\ y_i - f(x_i) \leq \varepsilon. \end{cases} \tag{5}$$

Constraints in equation (5) guarantee that the regression function $f(x)$ has at most an ε deviation from the actually obtained data point y_i for all training data. At the same time the minimization of the norm of the vector w requires the function $f(x)$ to be as flat as possible. The benefit of this condition will be seen in far more complex problems than fitting the linear function to an observed data.

Having introduced the problem (5) we can describe now the notion of feasibility. There are may be many vectors w and offsets b for which the function f satisfies the inequality constraints of (5). This set of parameters is called a feasible set. Hence, if the problem has this set of solutions then the problem is feasible.

The constrained problem, mentioned above, can be solved by the method of Lagrange multipliers. In further derivations we will use the following three theoretical results. First of all, we define Kuhn-Tucker Saddle Point theorem.

Theorem 1. *Assume an optimization problem of the form*

$$\min_x f(x)$$
$$\textit{subject to } c_i(x) \leq 0 \; \forall \; i \in [n], \tag{6}$$

where $f: \mathbb{R}^m \to \mathbb{R}$ *and* $c_i: \mathbb{R}^m \to \mathbb{R}$ *for* $i \in [n]$ *are arbitrary functions, and a Lagrangian*

$$L(x, \alpha) := f(x) + \sum_{i=1}^{n} \alpha_i c_i(x) \tag{7}$$

where $\alpha_i \geq 0$.

If a pair of variables $(\bar{x}, \bar{\alpha})$ *with* $\bar{x} \in \mathbb{R}^n$ *and* $\bar{\alpha} \geq 0$ *for all* $i \in [n]$ *exists, such that for all* $x \in \mathbb{R}^m$ *and* $\alpha \in [0, \infty)^n$ *we have*

$$L(\bar{x}, \alpha) \leq L(\bar{x}, \bar{\alpha}) \leq L(x, \bar{\alpha}),$$

then $\bar{x}$ *is a solution to (6) [5, p. 166].*

The function $f(x)$ in (6) to be minimized over the variable x is sometimes called the objective function.

Secondly, we introduce Karush-Kuhn-Tucker conditions for differentiable convex problems, which are necessary conditions for a solution to be optimal.

Theorem 2. *A solution to the optimization problem (6) with convex, differentiable* f, c_i *is given by* $\bar{x}$, *if there exists some* $\bar{\alpha} \in \mathbb{R}^n$ *with* $\bar{\alpha}_i \geq 0$ *for all* $i \in [n]$ *such that the following conditions are satisfied:*

$$\partial_x L(\bar{x}, \bar{\alpha}) = \partial_x f(\bar{x}) + \sum_{i=1}^{m} \bar{\alpha}_i \partial_x c_i(\bar{x}) = 0, \tag{8}$$

$$\partial_{\alpha_i} L(\bar{x}, \bar{\alpha}) = c_i(\bar{x}) \leq 0, \tag{9}$$

$$\sum_{i=1}^{m} \bar{\alpha}_i c_i(\bar{x}) = 0. \tag{10}$$

[5, p. 170]

Finally, another concept that is useful when dealing with optimization problems is that of *duality*. Define

$$g(\alpha) = \inf_x L(x, \alpha).$$

Due to the constraint of (6) and positivity of α_i we have for every feasible x the inequalities:

$$\sum_{i=1}^{n} \alpha_i c_i(x) \leq 0 => L(x, \alpha) \leq f(x) => g(\alpha) \leq f(x) => \max_{\alpha} g(\alpha) \leq f(x).$$

Therefore also the solution $\bar{x}$ of (6) satisfies

$$\max_{\alpha \geq 0} g(\alpha) \leq f(\bar{x}).$$

The problem of maximum

$$\max_{\alpha} g(\alpha)$$
$$\text{subject to } \alpha \geq 0 \tag{11}$$

is called a dual problem of (6).

In general in the case of a dual problem $\max_\alpha g(\alpha)$ might not always be equal to $f(\bar{x})$ (then it is said that there exists so called duality gap). But in case of convex optimization the following strong duality property holds:

Theorem 3. *If there exists x such that for all $i \in [n]$ $c_i(x) < 0$ (so called Slater's condition), then duality gap is zero:*

$$\max_{\alpha \geq 0} g(\alpha) = f(\bar{x}), \tag{12}$$

where $\bar{x}$ is a solution of (6).

This means that if Slater's condition is satisfied and functions f and c_i are convex, we can solve (6) as follows. First use (8) to eliminate primal variables x in the expression $L(x, \alpha)$, so that we obtain the function g. Then solve the dual problem and use (8) again to find the solution $\bar{x}$ of (6).

Let us apply the general theory to the problem (5). The Lagrange functional of (5) is:

$$L = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(\varepsilon + y_i - f(x_i)\right) - \sum_{i=1}^{m} \alpha_i^* \left(\varepsilon - y_i + f(x_i)\right)$$
$$\text{subject to } \alpha_i, \alpha_i^* \geq 0, i = 1, \dots, m. \tag{13}$$

where α_i and α_i^* are new variables (Lagrange multipliers) which have to satisfy positivity constraints.

Substitute (2) into (13) to get the following equation:

$$L = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(\varepsilon + y_i - \langle w, x_i \rangle - b\right) - \sum_{i=1}^{m} \alpha_i^* (\varepsilon - y_i + \langle w, x_i \rangle + b) \tag{14}$$

If the Slater's condition is satisfied (*i.e.* if there exists a regression hyperplane that is strictly within error bounds at all points x_i), this function has a saddle point with respect to the primal and dual variables at the solution [5, p. 254]. Hence, the above mentioned Kuhn-Tucker theorem could be used and finding the optimal solution of the problem (5) is equivalent to determining the saddle points of the Lagrange functional (14) [6, p. 252]. Thus, the partial derivatives of L are equated to zero:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0 \tag{15}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) x_i = 0 \tag{16}$$

Substituting (15) and (16) into (14) gives us:

$$L = \frac{1}{2} \langle \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) x_i \, , \sum_{j=1}^{m} \big(\alpha_j^* - \alpha_j\big) x_j \rangle - \sum_{i=1}^{m} \alpha_i \left(\varepsilon + y_i - \langle \sum_{j}^{m} \big(\alpha_j^* - \alpha_j\big) x_j \, , x_i \rangle - b \right)$$

$$- \sum_{i=1}^{m} \alpha_i^* \left(\varepsilon - y_i + \langle \sum_{j=1}^{m} \big(\alpha_j^* - \alpha_j\big) x_{j,} x_i \rangle + b \right) = \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i) \big(\alpha_j^* - \alpha_j\big) \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i \varepsilon$$

$$- \sum_{i=1}^{m} \alpha_i^* \varepsilon - \sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i^* y_i^* - \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i) \big(\alpha_j^* - \alpha_j\big) \langle x_i, x_j \rangle + \sum_{i=1}^{m} \alpha_i b - \sum_{i=1}^{m} \alpha_i^* b$$

$$= -\frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i) \big(\alpha_j^* - \alpha_j\big) \langle x_i, x_j \rangle - \varepsilon \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i (\alpha_i^* - \alpha_i)$$

Therefore, we can solve (5) by finding $\bar{\alpha}$ of the saddle point by solving the dual problem:

$$\max_{\alpha_i, \alpha_i^* \in \mathbb{R}^m} -\frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i) \big(\alpha_j^* - \alpha_j\big) \langle x_i, x_j \rangle - \varepsilon \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i (\alpha_i^* - \alpha_i) \tag{17}$$

$$\text{subject to } \textstyle\sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0.$$

The obtained above result is very useful since the training data appears in this maximization problem only as a dot products. It turns out that this fact enables us to generalize the result to many classes of functions f that depend on measurements x nonlinearly. Such generalizations are discussed in the next chapter.

In figure 1 (Appendix 1.1) is shown a simple dataset which consists of five sample points. According to optimization problem (5) the goal is to fit the regression function so that its

deviation from each of these data points is less or equal to ε. In this simple example ε = 0,2 and the bars, introduced in figure 1, represent ε deviation from data points. If we restrict the class of regression function to the linear case only, it is obvious from figure 1 that there is no possibility to find such straight line in order to satisfy all constraints of optimization problem (5).

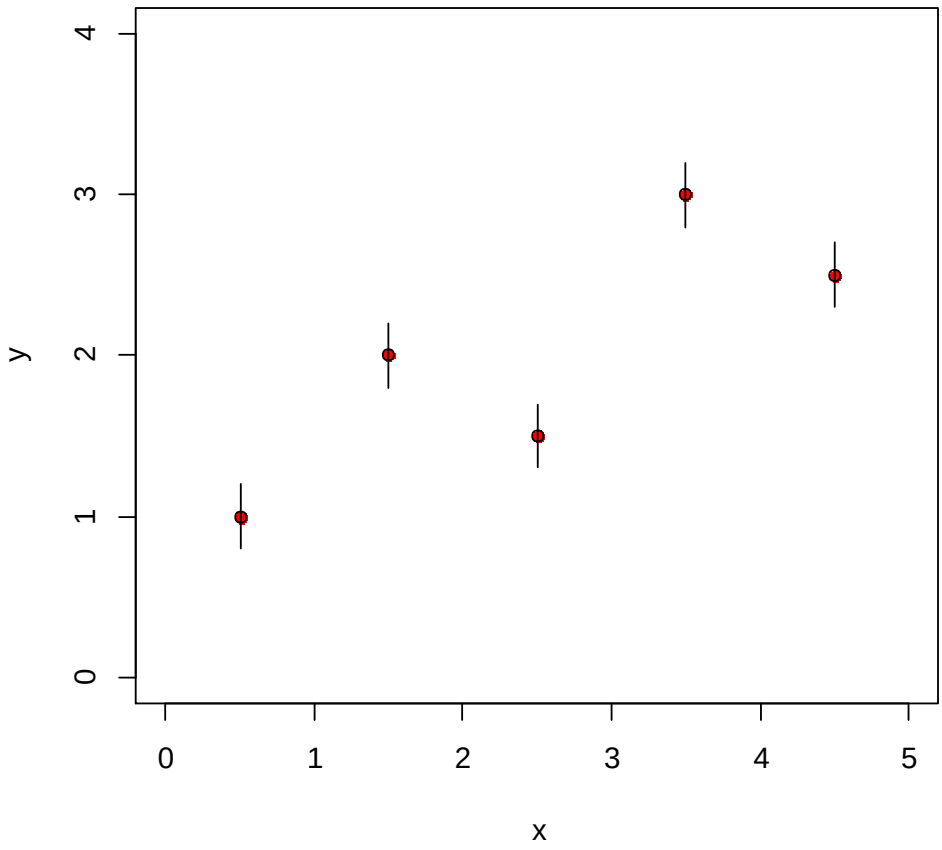


Figure 1. Sample points

This simple example, mentioned above, shows that the convex optimization problem (5) can be infeasible and it is important to be very careful with determining the value of ε. To make the example, explained above, feasible one can set ε parameter larger.

For instance, consider ε = 0.6. It is seen from figure 2, where the red and green lines represent two regression functions that satisfy all of the inequality constraints. The blue lines are the boundaries of the red line representing a tube with radius ε, so it is geometrically plausible that these data points have deviation from the red line less or equal to 0.6 and thus are fit in this tube.

Having changed ε to a larger value we made the feasible set nonempty. In other words, it is possible now to find many lines that will satisfy optimization constraints (5) and the red and green lines are just two options of the feasible set. The main difference between them is that the red line the flattest one among all possible regression lines satisfying the inequality constraints. It is the consequence of minimization of the norm of the vector w.

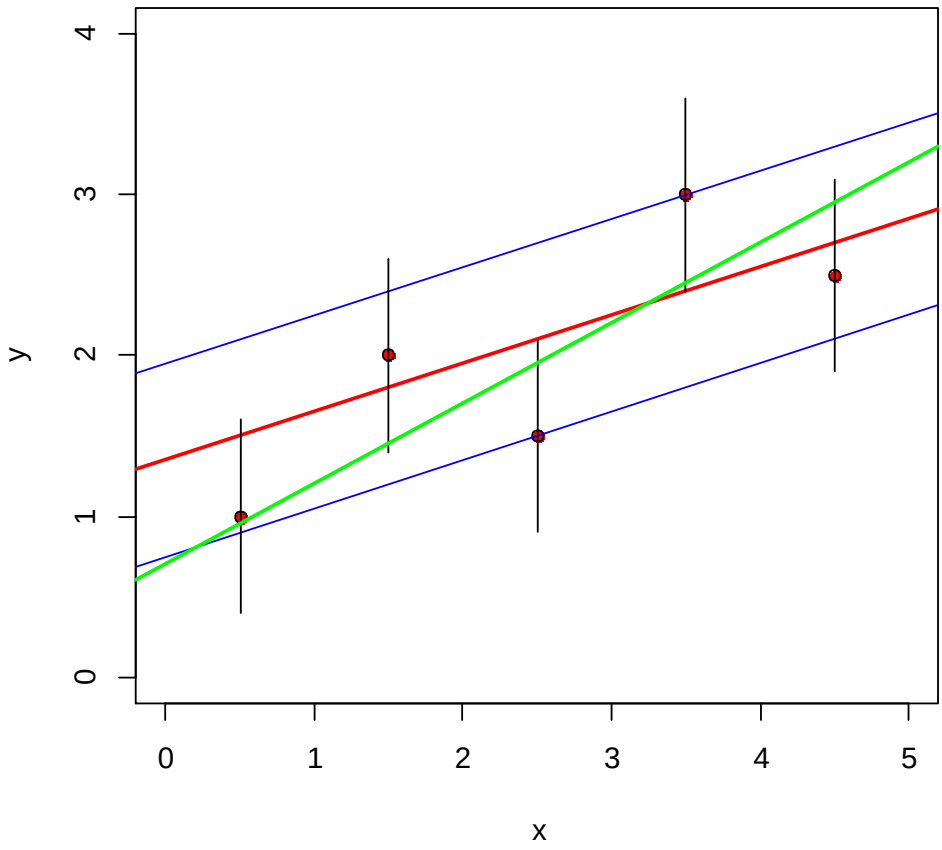


Figure 2. Two regression functions fitted to the data in the case of ε = 0.6

Such approach finds a solution to the optimization problem (5). Nevertheless, allowing all errors to be that large reduces the possibility of getting a good predictive model. Another approach is to allow errors to be larger than ε and to penalize the objective function for such terms.

To find a solution to optimization problem (5) in a context of our first example, shown in figure 1, one can introduce slack variables ξ_i and ξ_i^* as follows:

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$
$$\text{subject to} \begin{cases} f(x_i) - y_i \leq \varepsilon + \xi_i \\ y_i - f(x_i) \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0. \end{cases} \tag{18}$$

Now we don't try to strictly follow the constrains of (5), so the solution to otherwise infeasible problem can be found. Note that this problem is always feasible and the constraints satisfy Slater's condition as for sufficiently large values of ξ the all inequalities are strict. Figure 3 shows an optimal solution for our sample data in the case of ε = 0.2.

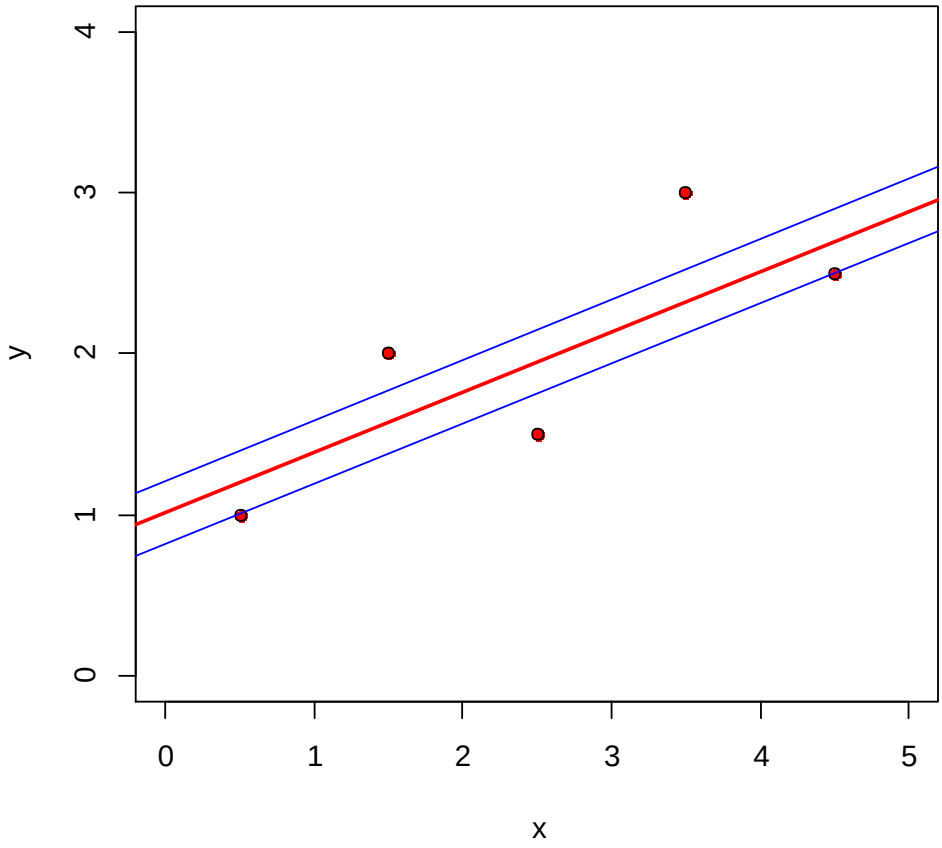


Figure 3. Fitted regression function in the case of ε = 0.2

Analogously to the procedure which was applied to optimization problem (5), one can use the method of Lagrange multipliers and the concept of duality for the problem (18). Hence, Lagrange function of optimization problem (18) looks as follows:

$$L = \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*) - \sum_{i=1}^{m}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \sum_{i=1}^{m}\alpha_i\left(\varepsilon + \xi_i + y_i - f(x_i)\right)$$
$$- \sum_{i=1}^{m}\alpha_i^*\left(\varepsilon + \xi_i^* - y_i + f(x_i)\right),$$

where Lagrange multipliers α_i, α_i^*, η_i, η_i^* have to satisfy positivity constraints.

Now the main difference between finding the saddle points of this Lagrange function and a Lagrangian (14) is that we get two extra partial derivatives:

$$\frac{\partial L}{\partial \xi_i} = \frac{C}{m} - \alpha_i - \eta_i = 0 => \eta_i = \frac{C}{m} - \alpha_i \quad \textbf{(19)}$$

$$\frac{\partial L}{\partial \xi_i^*} = \frac{C}{m} - \alpha_i^* - \eta_i^* = 0 => \eta_i^* = \frac{C}{m} - \alpha_i^*. \quad \textbf{(20)}$$

As α_i, α_i^*, η_i, $\eta_i^* \geq 0$ then from last derivations in (19) and (20) one obtains $\alpha_i \in [0, C/m]$, $\alpha_i^* \in [0, C/m]$.

Nevertheless, the dual problem for minimization problem (18) still remains the same as for the dual optimization problem (17), adding just one more constraint:

$$\max_{\alpha_i, \alpha_i^* \in \mathbb{R}^m} -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i^* - \alpha_i)\left(\alpha_j^* - \alpha_j\right)\langle x_i, x_j\rangle - \varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i^* - \alpha_i)$$
$$\text{subject to } \textstyle\sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0 \quad \textbf{(21)}$$
$$\alpha_i, \alpha_i^* \in [0, C/m].$$

Similary, as described in the case of dual problem (17) one can benefit from the fact that the training data appears in (21) only as dot products.

1.4 ε-SVM Regression

First of all, partial derivative (16) can be rewritten as

$$w = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)x_i.$$

By substituting it into (2) one obtains

$$f(x) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\langle x_i, x\rangle + b. \tag{22}$$

So the complete algorithm of $\boldsymbol{\varepsilon}$-SVM regression can be described in terms of dot products between the data. Solving (21) yields the values of α_i, α_i^* and x_i, which can be plugged into (22). When evaluating $f(x)$, we don't need to compute w explicitly. However, we still need to compute the value of b. For doing that we exploit Karush-Kuhn-Tucker (KKT) conditions described in Theorem 2, which state that the product between dual variables (α_i, α_i^*, η_i, η_i^*) and constraints in (18) has to vanish [5, 255 p.]:

$$\begin{aligned} \alpha_i\big(\varepsilon + \xi_i + y_i - f(x_i)\big) &= 0 \\ \alpha_i^*\left(\varepsilon + \xi_i^* - y_i + f(x_i)\right) &= 0 \end{aligned} \tag{23}$$

$$\begin{aligned} \eta_i\xi_i = 0 &=> (C/m - \alpha_i)\xi_i = 0 \\ \eta_i^*\xi_i^* = 0 &=> (C/m - \alpha_i^*)\xi_i^* = 0. \end{aligned} \tag{24}$$

If for some i in $\{1,2,\dots,n\}$ we have $\alpha_i \in (0, C/m)$ then the first equality constraint (24) yields $\xi_i = 0$. So we can easily find the offset b from the second factor of the first equation of (23). Concretely, using the definition (2) of f in the second factor of the first equation of (23), one computes b as follows:

$$b = y_i - \langle w, x_i\rangle + \varepsilon.$$

Same procedure applies in case $\alpha_i^* \in (0, C/m)$ for some i in $\{1,2,\dots,n\}$.

Unfortunately, this procedure is valid only if there exists an index i for which $\alpha_i \in (0, C/m)$ or $\alpha_i^* \in (0, C/m)$. In rare case when all α_i, α_i^* are either zero or C/m one needs to apply a different technique. We do not discuss it in the underlying thesis.

1.5 Support Vectors

Now we are at the crucial point, when we can explain the name of support vectors. First of all, suppose $\xi_i > 0$, then from (24) one obtains $\alpha_i = C/m$. Same applies for α_i^* when $\xi_i^* > 0$.

This means that data points with corresponding $\alpha_i = C/m$ (or $\alpha_i^* = C/m$) can lie outside the ε-insensitive tube around f.

Assume that $\alpha_i > 0$, then $\varepsilon + \xi_i + y_i - f(x_i) = 0$ in order to satisfy the constraint (23). That in turn implies $f(x_i) - y_i = \varepsilon$ when $\xi_i = 0$ and $f(x_i) - y_i > \varepsilon$ when $\xi_i > 0$. Same derivation applies for the second equation in (23). In other words, the Lagrange multipliers (α_i, α_i^*) may be nonzero only for $|f(x_i) - y_i| \geq \varepsilon$.

On the other hand, when $|f(x_i) - y_i| < \varepsilon$ this means that the Lagrange multipliers must be zero for the Karush-Kuhn-Tucker conditions to be satisfied, since slack variables ξ_i, ξ_i^* are positively defined. All in all, this means that only the points outside ε-insensitive tube around f and the points lying just right at the boundaries of this tube contribute to the function (22). Furthermore, these data points that come with nonvanishing Lagrange multipliers are called *Support Vectors*. It is geometrically plausible that the points inside the tube do not contribute to the solution: we could remove any of them, and still obtain the same solution, therefore they cannot carry any information about it [5, 256 p.].

2 Kernels

2.1 Data transformation to higher dimensional space

In the previous chapters the concept of regression estimation using support vectors was explained with theoretical and graphical examples. Though, if we want to fit a regression function that depends nonlinearly on the data values, then nonlinear transformation of data x_i to some higher dimension is useful. The key idea is to transform the sample points of data to a higher dimensional space and then apply a linear regression.

Suppose we have one dimensional feature x_i and we want to apply a transformation to $\mathbb{R}^2$ as follows:

$$x_i \rightarrow (x_i, x_i^2).$$

Further, using the definition of (2), one obtains:

$$f(x_i) = w_1 x_i + w_2 x_i^2 + b.$$

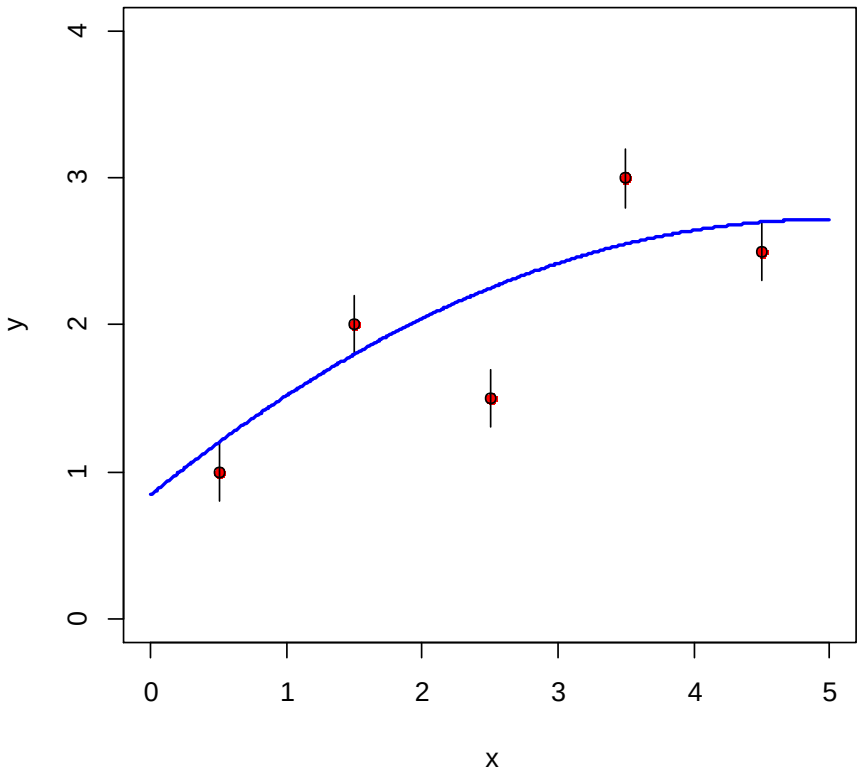


Figure 4. Quadratic function fitted to the data points

Thus we can easily use same ideas to fit a quadratic function of the data x instead of a linear function. The result of implementing of this idea in the case of our sample dataset can be seen in figure 4.

If the set of sample points is transformed into a dot product space by a transformation Φ, then the optimization problem (5) has the following form:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to} \begin{cases} f(\phi(x_i)) - y_i \le \varepsilon \\ y_i - f(\phi(x_i)) \le \varepsilon. \end{cases} \tag{25}$$

which has the dual problem:

$$\max_{\alpha_i, \alpha_i^* \in \mathbb{R}^m} -\frac{1}{2}\sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i)\left(\alpha_j^* - \alpha_j\right)\langle \phi(x_i), \phi(x_j)\rangle - \varepsilon \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i^* - \alpha_i) \quad \text{subject to } \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0 \tag{26}$$

Usually it is not required for the regression function to fit the sample points completely, even when the sample points have been transformed into the dot product space, because it often leads to overfitting the data - in the case of the training data estimated regression function will frequently behave well, though in case of the test set it produces results which are untrue.

If the set of transformed sample points does not conform with the constraints of the problem (25) or the constraints do not have to be filled then a cost term is added to this minimization problem as was done in the previous chapters.

Let the cost term be $\frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*)$, where the slack variables satisfy $\xi_i, \xi_i^* \ge 0$. Then the minimization problem looks as follows:

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*) \quad \text{subject to} \begin{cases} f(\phi(x_i)) - y_i \le \varepsilon + \xi_i \\ y_i - f(\phi(x_i)) \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, \end{cases} \tag{27}$$

where C is fixed positive constant.

Writing the given problem as a Lagrange functional and going through the same calculation steps as in chapter ("Quadratic Optimization Problem"), the following dual problem is obtained:

$$\max_{\alpha_i,\alpha_i^* \in \mathbb{R}^m} -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i^* - \alpha_i)\left(\alpha_j^* - \alpha_j\right)\langle\phi(x_i),\phi(x_j)\rangle - \varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i^* - \alpha_i)$$
$$\text{subject to } \textstyle\sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0 \qquad \textbf{(28)}$$
$$\alpha_i, \alpha_i^* \in [0, C/m].$$

The regression function $f(x)$ has the form:

$$f(x) = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)\langle\phi(x_i),\phi(x)\rangle + b. \qquad \textbf{(29)}$$

In order to adapt the theory described above to the set of nonlinear problems one needs to use the kernel function

$$K(x,y) \coloneqq \langle\Phi(x),\Phi(y)\rangle.$$

As the training data appears in dual optimization problem (28) and regression function (29) only as dot products there is no need to explicitly map all training inputs to some higher dimensional feature space. It is required to know the kernel, but not the space that gives us the value of the previously mentioned scalar product. For example, the transformation

$$x_i \to (x_i, x_i^2)$$

leads to the kernel

$$K(x,y) = xy + (xy)^2.$$

Note that we have to know only the kernel corresponding to the transformation used, knowing other details of the transformation is not necessary for applying the theory. By using the kernel the regression function takes the form:

$$f(x) = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)K(x_i,x) + b. \qquad \textbf{(30)}$$

2.2 Classification of Kernels

There is a number of suitable kernel functions. We shall give a short overview of the best known ones.

1. The *linear kernel* is the simplest kernel function. It is given by the inner product with an optional constant c added.

$$K(x, y) \coloneqq \langle x, y \rangle + c.$$

2. The *Gaussian kernel*

$$K(x, y) \coloneqq e^{\frac{-\|x-y\|^2}{2\sigma^2}} = e^{-\gamma \|x-y\|^2}.$$

is a good example of radial basis function kernel. The term "radial basis function" means that the kernel depends on x and y only through the euclidean distance between x and y. In figure 4 is shown the Gaussian kernel with $\gamma = 0.7$ applied to example explained in the chapter "Quadratic optimization problem".

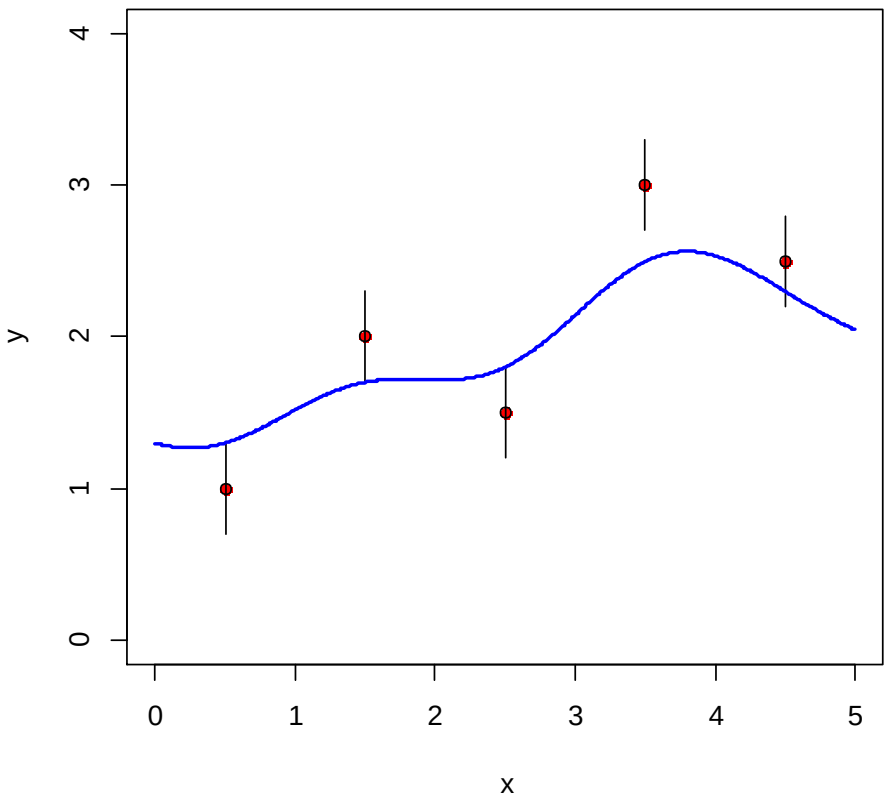


Figure 5. Radial Basis Function with gamma = 0.7, epsilon = 0.3

It is seen in figure 5 that the first four data points regarded from the left are the support vectors. The fifth data point is not a support vector and does not make any contribution to the regression function depicted as a blue line in the examined figure. The reason for such classification of these sample points follows from the inequality $|f(x_i) - y_i| < \varepsilon$ described in the chapter "Support Vectors".

The parameter γ (or σ) determines the width of the Gaussian kernel. One must be careful not to under- or overestimate the value of γ (or σ). If underestimated (in case of σ - overestimated), the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. If γ is set to be smaller than in previous figure, the regression function flattens and looks more like a linear function. It is useful to mention that the support vectors in figure 6 are the first, the third and the fourth sample points regarded from the left.

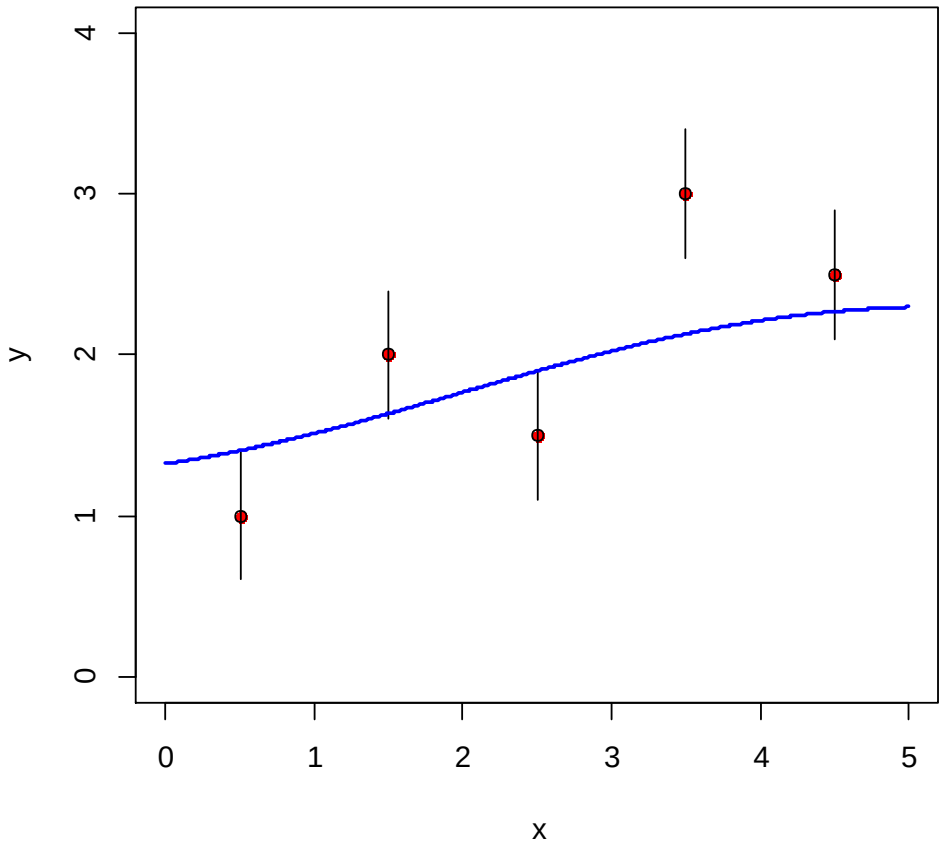


Figure 6. Radial Basis Function with gamma = 0.05, epsilon = 0.4

On the other hand, if γ is overestimated (for σ - underestimated), the function will lack regularity and the regression function will be highly sensitive to noise in training data. This is the consequence of the low value of σ which implies the fact that the width of the Gaussian curves is small as well. The regression function for that case is shown in figure 7. Similarly as before it is clear which sample points are the support vectors; the support vectors are all sample points except the second one regarded from the left.

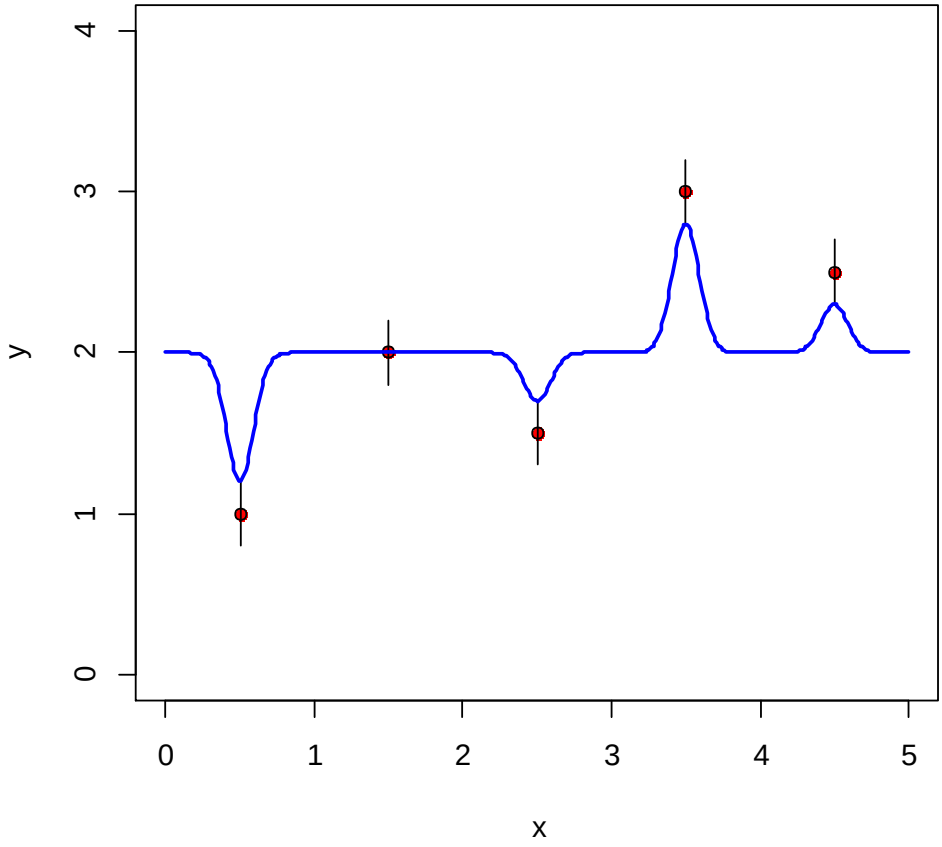


Figure 7. Radial Basis Function with gamma = 60, epsilon = 0.2

3. The *polynomial kernels* are widely used in cases where the training data is normalized. The slope α, the constant term c and the polynomial degree are adjustable:

$$K(x, y) \coloneqq (\alpha\langle x, y\rangle + c)^{degree} .$$

4. The *hyperbolic tangent kernel* is also known as the *sigmoid kernel* and looks as follows:

$$K(x, y) \coloneqq \tanh(\alpha\langle x, y\rangle + c),$$

where are two adjustable parameters, the slope α and the intercept constant c.

2.3 ν-SVM Regression

The parameter ε of the ε-insensitive loss is useful if the desired accuracy of the approximation can be specified beforehand. In some cases, however, we just want the estimate to be as accurate as possible, without having to commit ourselves to a specific level of accuracy a priori. We now describe a modification of the ε-SVR algorithm, called ν-SVR, which automatically computes ε [5, p. 260].

The main difference of ν-SVR from ε-SVR is that we add a new term ν mentioned to penalize the error term ε.

So the minimization problem of ν-SVR looks as follows:

$$\min_{w,b.\varepsilon,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + C(\nu\varepsilon + \frac{1}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*))$$
$$\text{subject to} \begin{cases} f(x_i) - y_i \le \varepsilon + \xi_i \\ y_i - f(x_i) \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, \varepsilon \ge 0. \end{cases} \quad \textbf{(31)}$$

Similarly to chapter "Quadratic optimization problem" the Lagrange function is obtained:

$$L(w, b, \alpha_i, \alpha_i^*, \beta, \xi_i, \xi_i^*, \varepsilon, \eta_i, \eta_i^*) = \frac{1}{2}\|w\|^2 + C\nu\varepsilon + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*) - \beta\varepsilon$$
$$- \sum_{i=1}^{m}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \sum_{i=1}^{m}\alpha_i\left(\varepsilon + \xi_i + y_i - f(x_i)\right) - \sum_{i=1}^{m}\alpha_i^*\left(\varepsilon + \xi_i^* - y_i + f(x_i)\right),$$

where $\alpha_i, \alpha_i^*, \beta, \eta_i, \eta_i^* \ge 0$ are Lagrange multipliers.

Further, setting the derivatives with respect to the primal variables equal to zero yields the five equations:

$$\sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0$$

$$w = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)x_i$$

$$\frac{C}{m} - \alpha_i - \eta_i = 0$$

$$\frac{C}{m} - \alpha_i^* - \eta_i^* = 0$$

$$Cv - \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) - \beta = 0.$$

Substituting the above five conditions into Lagrange functional leads to the dual optimization problem. We will state it in the kernelized form:

$$\max_{\alpha_i, \alpha_i^* \in \mathbb{R}^m} -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i^* - \alpha_i)\left(\alpha_j^* - \alpha_j\right)K(x_i, x_j) + \sum_{i=1}^{m} y_i(\alpha_i^* - \alpha_i)$$
$$\text{subject to } \textstyle\sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0$$
$$\alpha_i, \alpha_i^* \in [0, C/m] \qquad \textbf{(32)}$$
$$\sum_{i=1}^{m}(\alpha_i - \alpha_i^*) \leq Cv.$$

The regression estimate then takes the form:

$$f(x) = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)K(x_i, x) + b.$$

2.4 R Packages

In the underlying thesis shall be used a free software programming language R. There are many implementations of Support Vector Machines in R. Among them it is possible to find packages *e1071*, *kernlab*, *klaR* and *svmpath*. In this work is used package *kernlab* for making the examples with figures and package *e1071* for the price prediction as it performs better in terms of training time than the other packages. For comparison of these four SVM implementations see [7].

2.4.1 Kernlab

The *ksvm()* function, *kernlab*'s implementation of SVMs, is mostly written in R but uses libraries *bsvm* and *libsvm* which provide a very efficient C++ version of the Sequential Minimization Optimization (SMO). For regression, *ksvm()* includes the ε-SVM regression algorithm along with the ν-SVM regression formulation [7].

The choice of kernels in *kernlab* package is rich. Among others the most popular kernel functions which can be used by setting the kernel parameter to the following value:

- The linear kernel (set the parameter kernel to *kernel = "vanilladot"*)
$$K(x, y) = \langle x, y \rangle$$
- The polynomial kernel (*kernel = "polydot"*)
$$K(x, y) = (scale\langle x, y \rangle + offset)^{degree}$$
- The Gaussian RBF kernel (*kernel = "rbfdot"*)
$$K(x, y) = e^{-\sigma \|x-y\|^2}$$
- The hyperbolic tangent kernel (*kernel = "tanhdot"*)
$$K(x, y) = \tanh(scale\langle x, y \rangle + offset)$$

In addition, in the case of a Gaussian RBF kernel function parameter *kpar* can be set to the string *"automatic"*, which tries to find an optimal σ itself. The typical command to fit the model could look as follows:

svm.model = ksvm(x, y, scaled = F, type = "eps-svr", kernel = "rbfdot", kpar = "automatic"), where

- *x* – a matrix or vector containing the training data;
- *y* – a numeric response vector;
- *scaled* – a logical vector indicating the variables to be scaled;
- *type* – SV regression type;
- *kernel* – the kernel function used in training and predicting;
- *kpar* – a list of the parameters to be used with the kernel function.

It is useful to mention that package *kernlab* allows defining custom kernels (so called user-defined kernels).

2.4.2 e1071

svm() function of the package *e1071* uses the library *libsvm*. The function *svm()* allows to use the ε-SVM regression and ν-SVM regression as well as *kernlab*'s *ksvm()* function. It has less types of kernels available, but still there are the most known ones:

- The linear kernel (set the parameter kernel to *kernel = "linear"*)
$$K(x, y) = \langle x, y \rangle$$
- The polynomial kernel (*kernel = "polynomial"*)
$$K(x, y) = (\gamma \langle x, y \rangle + coef0)^{degree}$$
- The Gaussian RBF kernel (*kernel = "radial"*)
$$K(x, y) = e^{-\gamma \|x-y\|^2}$$
- The hyperbolic tangent kernel (*kernel = "sigmoid"*)
$$K(x, y) = \tanh(\gamma \langle x, y \rangle + coef0)$$

In order to fit the SVM model one can execute the following command:

svm.model = svm(y ~ . , data = variables, scale = F, type = "eps-regression", kernel = "radial"), where

- *y* ~ . – formula describing the model;
- *data* – data frame containing the variables required for fitting the models;
- *scale* – a logical vector indicating the variables to be scaled;
- *type* – SV regression type;
- *kernel* – the kernel function used in training and predicting.

2.5 Cross-validation

Cross-validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against [8]. There are many types of different implementations of cross-validation. In this work is used *R* repeated *K*-fold *(R* x *K)* cross-validation. That is, the original sample is randomly partitioned into *k* equal size subsamples. Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining *k-1* subsamples are used as training

data. The cross-validation process is then repeated *k* times, with each of the *k* subsamples used exactly once as the validation data [9]. Finally, the procedure described above repeated *R* times. As the result, we get *R*k* estimations that can be averaged to produce a single estimation.

In the underlying thesis is used an implementation of cross-validation from package *cvTools*. We use *cvTuning()* function from this package to evaluate the best parameters of SVM regression model via repeated *K*-fold cross-validation.

The list of desired parameters can be submitted to the main function if we define the following variable:

Tuning = list(gamma = c(0.05, 0.01), cost = c(1, 2, 4, 8)),

where *gamma*, *cost* are chosen parameters to be fit to the model.

Further, we include this variable into the function *cvTuning()* along with other arguments, such as:

- *object* – a function for fitting a model (*i.e. svm()* function);
- *formula* – formula describing the model;
- *data* – a data frame containing the variables required for fitting the models;
- *tuning* – a list of arguments giving the tuning parameter values to be evaluated (i.e *Tuning* variable);
- *cost* – a cost function measuring prediction loss;
- *K* – an integer giving the number of groups into which the data should be split;
- *R* – an integer giving the number of replications for repeated *K*-fold cross-validation.

To apply repeated *K*-fold cross-validation using the package *cvTools* one can use:

svm.tuning = cvTuning(svm, formula = y ~ ., data = variables, tuning = Tuning, cost = rmspe, K = 10, R = 10).

3 Prediction of stock price

In order to illustrate the theoretical part introduced in previous chapters, the high-frequency stock data of the British multinational telecommunications company Vodafone Group Plc and the British-Swedish multinational pharmaceutical and biologics company AstraZeneca Plc is being used in the following analysis. In this thesis we consider the problem of predicting future stock prices for a short time period by using the market data available at the beginning of the prediction interval. If the price movements can to certain extent be predicted, then such predictions can be used in various trading algorithms. The implementations of the predictions in trading algorithms are not being considered as it is not relevant to the underlying thesis.

3.1 Order types

An order in a market is an instruction from customers to brokers to buy or sell securities on the exchange [10]. There are different types of orders. In this particular case we distinct *market orders* and *limit orders*. Market orders are orders to buy or sell securities at the best available price. Limit orders are orders to buy or sell securities at a particular price [11, p. 61]. The whole high-frequency data can be referred as *limit order book* which holds information about unexecuted limit orders and constantly updates it in case a new limit order is added or cancelled. Market orders are not represented in limit order book itself. However, they can be tracked by trade executions.

Nowadays the limit order books are completely electronic. Thus, computers register all the updates automatically. In case the investor wants to get the most recent updates of the limit order book, one has to pay for the data frame file. High-frequency data features are following:

- *date* – date of trading day;
- *sym* – a financial security identification code;
- *localtime* – local time of the updates;
- *bid1, bid2 etc* – best buying prices available at a particular moment;
- *ask1, ask2 etc* – best selling prices available at a particular moment;
- *bsize1, bsize2 etc* – available amount of shares for bid1, bid2 etc respectively;
- *asize1, asize2 etc* – available amount of shares for ask1, ask2 etc respectively;
- *ordersb1, ordersb2 etc* – number of orders for bid1, bid2 etc respectively;

- *ordersa1, ordersa2 etc* – number of orders for ask1, ask2 etc respectively;
- *is_a_trade* – shows whether trade execution took place or not;
- *price* – the price, or quote, of the last trade execution;
- *size* – size, or volume, of sold or bought securities during the last trade execution;
- *tradetime* – the time for the last trade execution.

Example of the high-frequency data frame is in Appendix 2.

3.2 VWAP

Having such various high-frequency data, it is necessary to designate what value we want to predict. In current thesis is used a *volume-weighted average price* (VWAP). The reason for such choise is an opportunity to compare the results with master's thesis of Andrei Orlov [1] where he used VWAP as a high-frequency price. VWAP stands for the average quote per share of all trades of the certain period of time. It can be represented by the following formula:

$$VWAP_i = \frac{\sum_t P_{it} V_{it}}{\sum_t V_{it}}, \quad \{t\} \in T$$

where V_{it} is the volume of trade i executed at time interval t, P_{it} is the price of trade i at time interval t, T is a trading day.

The actual predictions are made not for VWAPs but for the increments between them:

$$inc = VWAP_{i+1} - VWAP_i.$$

3.3 Regressors

As it was mentioned in chapter Loss functions, in regression analysis one can subdivide variables into two classes: *independent* and *dependent variable*. Independent variable, that is called *regressor*, has some influence to dependent variable, and this connection helps to make predictions. In our case the dependent variable is an increment between two VWAPs. The main task is to find the set of regressors that helps us to forecast the dependent variable as well as possible. For our SVM model the following variables were extracted:

- market order intensity – average number of market orders executed in a unit of time, calculated for ask and bid side of limit order book;
- buy and sell pressure – a sum of exponentially weighted volumes of limit orders on the bid and ask sides respectively;
- limit order intensity – average number of limit orders added to bid1 and ask1 queue in a unit of time;
- cancellation intensity – average number of cancelled limit orders in bid1 and ask1 in a unit of time;
- difference between period last trade price and period VWAP value.

One needs to be careful with adding new regressors to already available. A new independent variable may improve the fitted model if it has a low correlation with other regressors. Otherwise it makes the model more complex and inefficient in terms of training time.

In order to compare different methods of estimation in the case of a specific data set, some type of average prediction error is calculated for predictions. There are several types available, but in this work is considered root mean square error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2},$$

where y_i is actual observed value and $f(x_i)$ is a value predicted by a regression model.

The obtained indicator gives an idea of how well the observed data can be predicted for certain methods. However, the calculated indicator must be treated with caution, especially if the parameters of the method have been found using the same data as for calculating the indicators (it is possible to find coherent parameters for the data, but they might result in unreasonable predictions in the case of new data, so called overfitting effect) [12, p. 22].

3.4 Description Of Algorithms

In this subsection we will describe how to use the primary algorithm for regression analysis of stock price. Further, we briefly introduce the secondary algorithm in order to compare the main algorithm.

3.4.1 Primary Algorithm

The prediction algorithm is an example of the SVM theory implementation in R. In order to use this algorithm it is necessary to install packages *e1071* and *cvTools*, described in the chapters "R packages" and "Cross-validation". Before executing the algorithm it is possible to change the values of tuning parameters γ and C that would be used in fitting ε-SVR model with the sigmoid kernel to a data.

For forecasting the stock prices one needs to provide the following four arguments to the function *PricePrediction()*:

- *Pre_Calc_Days* – the number of days for calculating just regressors;
- *Limit_Days* – the number of days to store the regressors;
- *Calc_Days* – the number of days to make forecasts;
- *PERIOD_LENGTH* – the length of a single trading period.

Once the arguments are passed to the function it calculates the regressors, described earlier in this work, at the end of every trading period except the last one. The same actions are performed for finding VWAP values except that the algorithm drops VWAP value of the first period of a day. Then the system finds the increments between VWAPs. This procedure is repeated as many times as defined in *Pre_Calc_Days* and the data is stored in the memory. This is the end of the first stage.

Further, in the second stage of the algorithm at the beginning of every day cross-validation is applied in order to find the best parameters for the model fitting. Once cross-validation is done the algorithm fits ε-SVR model to a data which is valid till the end of the day. Similarly to the first stage, regressors, VWAPs and their increments are found throughout the periods of a day. In addition, forecasts of increments are made and predicted VWAPs calculated.

Note that if the number of days exceeds the value of *Limit_Days* the algorithm deletes outdated regressors from the memory. The connections between various functions of algorithm code are shown in Appendix 3.

3.4.2 Secondary Algorithm

The prediction algorithm was selected from the master's thesis of A. Orlov (2012) [1]. The idea of this algorithm is rather similar except that a different econometric model is fitted to

the data. In his work Orlov used linear regression analysis and tried to improve the results by applying autoregressive model to residuals of linear regression model. Unlike the primary algorithm the secondary one tries to make predictions to arithmetic rate of returns (ROR) of *VWAPs* that could be defined in terms of $VWAPs$ as follows:

$$VWAP_i = \frac{VWAP_{i+1} - VWAP_i}{VWAP_i}.$$

The algorithm provides day splitting into intervals as well. This allows using different models for different intervals. Though, it did not give any significant advantage or disadvantage to price prediction process. So the comparison of the primary algorithm is done to the secondary algorithm, which does not use the splitting.

3.5 Parameter fitting

Using cross-validation technique for tuning the parameters of SVM model is quite expensive in terms of training time to fit all possible combinations of parameters. We perform some tests to decide which kernel to use and to choose a pair of values for C and γ parameters, which will later be used in cross-validations when we choose a model for each new training day.

In order to find the best parameters we fit ε-SVR model to data of AstraZeneca stock. The day is divided into 15 minute intervals, the first 9 days are selected for finding regressors, 5 days are picked to predict the prices and all new regressors obtained after the first 9 days are used in the further calculations. As regressors are collected, at the beginning of each day we fit ε-SVR model to the data. First of all, we will search for the best parameters of the polynomial kernel with degree = 3, modifying the values of C and γ. The parameter ε value is always 0.1. For accuracy measures we use the percentage of correctly computed stock price movement directions and root mean square error. In Table 1 are presented the results of 7 experiments.

Experiment	1	2	3	4	5	6	7
C	1	1	1	2	0.1	1.5	0.5
γ	0.1	0.05	0.01	0.05	0.05	0.05	0.05
Accuracy	64.24%	63.03%	52.12%	62.42%	57.58%	62.42%	62.42%
RMSE	3.676	3.511	3.542	3.541	3.45	3.577	3.475

Table 1. Test results of the polynomial kernel in the case of AstraZeneca stock

In order to conclude whether the polynomial function fits for predicting the stock price one must compare it with test results of another kernel. So the same procedure is made for the Gaussian kernel.

Experiment	1	2	3	4	5	6	7
C	1	1	1	1	2	2	1.5
γ	0.1	0.05	0.01	0.005	0.01	0.005	0.005
Accuracy	67.88%	64.85%	69.09%	69.7%	70.3%	69.09%	69.09%
RMSE	2.814	2.796	2.707	2.736	2.725	2.707	2.715

Table 2. Test results of the Gaussian kernel in the case of AstraZeneca stock

As it is seen from Table 1 and 2 the SVM algorithm that uses the Gaussian kernel predicts the stock price movement direction better than polynomial. If we measure the results in terms of root mean square error, the difference will be rather bigger in favor of the Gaussian kernel. Also it is seen from Table 2 that the range of values between 0.01 and 0.005 can be a good choice for the parameter γ. Before making the final decision in favor of the Gaussian kernel we introduce the test results for the sigmoid kernel.

Experiment	1	2	3	4	5	6	7
C	1	1	1	1	1	2	2.5
γ	0.1	0.05	0.01	0.005	0.008	0.01	0.01
Accuracy	49.09%	60%	70.91%	69.09%	69.7%	72.12%	70.91%
RMSE	23.693	7.704	2.73	2.816	2.752	2.704	2.692

Table 3. Test results of the sigmoid kernel in the case of AstraZeneca stock

The test results of sigmoid kernel are rapidly improving as parameters C and γ are changed. Such improved results can compete with the test results of Gaussian kernel. As the result, the best values of sigmoid kernel have some advantage before the values of Gaussian kernel. Therefore, in further calculations of AstraZeneca stock a sigmoid kernel is chosen.

Now let us make the same procedure in the case of the data of Vodafone stock. As before we use 15 minute intervals, 9 days for finding the primary regressors, 5 days for predicting the prices and all new regressors obtained after the first 9 days are used in the further calculations. The accuracy results of the polynomial in the case of Vodafone stock is seen in Table 4.

Experiment	1	2	3	4	5	6	7
C	1	1	1	2	0.1	1.5	0.5
γ	0.1	0.05	0.01	0.05	0.05	0.05	0.05
Accuracy	66.06%	66.67%	50.3%	64.85%	60%	66.67%	64.24%
RMSE	0.183	0.172	0.183	0.173	0.176	0.173	0.172

Table 4. Test results of the polynomial kernel in the case of Vodafone stock

If we fit ε-SVR model with a Gaussian model to the same data of Vodafone stock, then the results of the prediction accuracy look as follows.

Experiment	1	2	3	4	5	6	7
C	1	1	1	1	2	2	3
γ	0.1	0.05	0.01	0.005	0.01	0.005	0.005
Accuracy	66.67%	69.09%	70.91%	68.48%	70.3%	70.91%	70.91%
RMSE	0.161	0.158	0.151	0.152	0.151	0.15	0.15

Table 5. Test results of the Gaussian kernel in the case of Vodafone stock

In Table 6 are presented the testing results for the sigmoid kernel.

Experiment	1	2	3	4	5	6	7
C	1	1	1	1	2	3	4
γ	0.1	0.05	0.01	0.005	0.01	0.01	0.01
Accuracy	52.12%	57.58%	66.67%	67.88%	67.27%	68.48%	67.88%
RMSE	1.186	0.397	0.151	0.154	0.151	0.151	0.153

Table 6. Test results of the sigmoid kernel in the case of Vodafone stock

In the case of Vodafone stock the testing results reveal that the Gaussian kernel and the sigmoid kernel are the better choice than the polynomial kernel. In terms of root mean square error the best testing results are rather similar. The Gaussian kernel has a small advantage in terms of the percentage of correctly computed stock price movement directions. Hence, we shall use the Gaussian kernel in order to compute the predictions of the price of Vodafone stock.

Once the kernel functions are chosen for two stocks it is possible to test the primary algorithm versus the secondary algorithm. Now we will consider AstraZeneca stock and 3 different time

intervals. Both algorithms use 9 days for collecting regressors and the predictions are made for 84 days. In the case of the primary algorithm only 15 days are used to store regressors values and sigmoid kernel is chosen with $\varepsilon = 0.1$. In addition, after computing initial regressors on the basis of the first 9 days the primary algorithm uses 4 times repeated 5-fold cross-validation technique for tuning parameters. The parameter γ is chosen between values 0.01 and 0.008. The value of C can be whether 1 or 2. As soon as cross-validation tool picked the best combination of γ and C ε-SVR model is applied to the data of AstraZeneca stock. The comparison of the primary and the secondary algorithms is seen from Table 7.

Stock	AstraZeneca		
Period length	5 min	10 min	15 min
The secondary algorithm	72.2%	72.49%	72.76%
	2.1	2.816	3.314
The primary algorithm	72.05%	72.04%	72.58%
	2.142	2.869	3.311

Table 7. Comparison between the primary and the secondary algorithms

In the case of Vodafone stock the same procedure is applied as described above except that for the SVM model fitting the Gaussian kernel is used. The parameter γ is picked between values 0.01 and 0.005. The value of C is still 1 or 2. The accuracy results of the primary and the secondary algorithms in the case of Vodafone stock are presented below.

Stock	Vodafone		
Period length	5 min	10 min	15 min
The secondary algorithm	67.11%	69.02%	69.19%
	0.118	0.152	0.178
The primary algorithm	67.47%	69.45%	70.24%
	0.119	0.153	0.175

Table 8. Comparison between the primary and the secondary algorithms

Summary

The aim of the underlying thesis was to implement support vector machines (SVM) on the short-term price movement prediction in intra-day trading. In order to research that topic the predictive algorithm was developed. The data of two multinational companies were used in the testing of the predictive algorithm.

Finding a model that is able to predict a stock price is a very serious challenge. The predictive algorithm, developed in the underlying thesis, uses SVM, a recently popular method of machine learning. We have demonstrated that with support vector machines it is possible to get as good results as with other approach with quite small modeling effort. One drawback of SVM approach is the slowness of the cross-validation process. One of the interesting open questions remaining is the optimal choice of how often and how to perform extensive cross-validations so that the computation speed is high and the accuracy does not suffer.

In addition to the development of the predictive algorithm, it was also given an insight into the theory of support vector regression. The concept of regression estimation was explained with theoretical and graphical examples. The possibility to use non-linear regression functions through the kernel functions in a framework of SVM was introduced as well. The further improvements of the predictive algorithm may involve the minimization of the training time and the research of dependencies of errors. Additionally the next step could be the implementation of the predictions to trading algorithms.

Tugivektormasinate kasutatavuse uurimine lühiajaliste hinnaliikumiste prognoosimiseks päevasisese kauplemise tingimustes.

Vassili Mušnikov

Kokkuvõte

Kaasaegses finantsmaailmas on väga populaarsed automaatkauplemise algoritmid. Automaatse kauplemise tingimustes on sageli oluline osata prognoosida hinnaliikumise suundi ja ulatust suhteliselt lühikese aja, näiteks 10 minuti jooksul. On mitmeid statistilisi meetodeid selle ülesanne lahendamiseks, kuid käesoleva magistritöö raames on kasutatud üks kaasaegne meetod – tugivektormasinad. Töö eesmärgiks on uurida meetodi toimimist meid huvitava andmestiku põhjal, milleks on Vodafone-i ja AstraZeneca andmed.

Töö koosneb kolmest osast. Esimene peatükk on pühendatud tugivektor masinatele. Selleks, et paremini aru saada selle meetodi töö põhimõtetest, seletatakse töös teatud aluseid. Esimeses peatükis räägitakse sellest, mis on tehisõpe, selgitatakse, kuidas leida regressiooni funktsiooni tugivektor masinate abil ja mainitakse probleeme, mis võivad tekkida regressiooni funktsiooni leidmisel. Regressiooni funktsiooni kontseptsioonid on selgitatud nii teoreetiliste kui graafiliste näidete abil.

Teine peatükk on pühendatud tuumafunktsioonidele, kuna need võimaldavad kasutada mittelineaarseid funktsioone kui regressiooni funktsioone. Ühtlasi selgitakse lugejale, mille poolest tuumafunktsioonide kasutus lihtsustab regressiooni funktsiooni leidmise. Lisaks tutvustatakse lühidalt teises peatükis programmeerimiskeele R lisamoodulite tehnilisi võimalusi. Lõpuks, mainitakse lühidalt sellise statistilise meetodi algoritmide võrdlemiseks nagu rist-validatsioon.

Erinevalt kahest esimesest peatükist, mis annavad teoreetilise ülevaate, kannab kolmas peatükk praktilise iseloomu. Peatükk tutvustab, kuidas tugivektor masinaid rakendatakse lühiajaliste hinnaliikumiste prognoosimiseks päevasisese kauplemise tingimustes. Selles peatükis selgitatakse hindade ennustamise algoritmi ning kirjeldatakse samuti kasutatavid andmeid. Seoses mõningate sarnasustega, toob autor välja ka oma töö võrdluse Andrei Orlovi magistritööga. Lisaks lugeja leiab töö lõpus lisad, mis koosnevad andmestikust, joonisest, mis selgitab koodi funktsioonide vahelisi suhteid, jooniste koode ja CD, mis sisaldab algoritmi koodi.

Töö autor arendas hindade ennustamise algoritmi, mis põhineb tugivektor masinatel. Ühtlasi demonstreeris töö autor, et tugivektor masinate abil on võimalik saada sama häid tulemusi nagu ka teiste lähenemiste puhul. Üheks tugivektor masinate meetodi puuduseks on selle aeglus rist-validatsiooni protsessis. Jääb avatuks küsimus, kui tihti ja kuidas tuleb teostada rist-validatsioone, et arvutused oleksid kiired ja täpsus ei kannataks. Ennustava algoritmi edaspidine arendamine võib olla seotud arvutuste aja minimiseerimisega ja vigade sõltuvuse uurimisega. Lisaks järgmine samm võib olla ennustuste rakendamine kauplemise algoritmides.

Appendix 1

Appendix 1.1

```
# sample data points
x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)
# plot the data
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
eps = 0.2
# epsilon deviation from data points
for(i in 1:length(x)){
arrows(x[i],y[i]-eps,x[i],y[i]+eps,length=0,code=3,angle=90)
}
```

Appendix 1.2

```
library(kernlab)

x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
eps = 0.6
for(i in 1:length(x)){
    arrows(x[i],y[i]-eps,x[i],y[i]+eps,length=0,code=3,angle=90)
}
# fit the epsilon support vector regression
eps.svr <- ksvm(x,y, scaled = FALSE, type = "eps-svr", kernel = "vanilladot", C = 10, epsilon=eps)
# make predictions
esvr.pred <- predict(eps.svr,x)
# solid line for espilon-svr predictions
fit <- lm(esvr.pred~x)
abline(fit,col="red",lwd =2)
# boudaries of fitted regression model
abline(fit$coefficients[1] - eps, fit$coefficients[2], col="blue")
abline(fit$coefficients[1] + eps, fit$coefficients[2], col="blue")
# another possible regression function
abline(0.7,0.5,col="green", lwd = 2)
```

Appendix 1.3

```
library(kernlab)

x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
eps = 0.2
# fit the epsilon support vector regression
eps.svr <- ksvm(x,y, scaled = FALSE, type = "eps-svr", kernel="vanilladot", C = 10, epsilon=eps)
esvr.pred <- predict(eps.svr,x)
# solid line for espilon-svr predictions
fit<- lm(esvr.pred~x)
abline(fit,col="red",lwd =2)
# boundaries of fitted regression function
abline(fit$coefficients[1] - eps, fit$coefficients[2], col="blue")
abline(fit$coefficients[1] + eps, fit$coefficients[2], col="blue")
```

Appendix 1.4

```
library(kernlab)

x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)
eps = 0.2
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
for(i in 1:length(x)){
arrows(x[i],y[i]-eps,x[i],y[i]+eps,length=0,code=3,angle=90)
}
# user-defined kernel
k <- function(x,y) {x*y + (x*y)**2}
class(k) <- "kernel"
# fit the epsilon support vector regression
eps.svr <- ksvm(x,y, scaled = FALSE, kernel=k, C=10,epsilon = eps)
# make predictions
x1<- seq(0,5,0.01)
esvr.pred <- predict(eps.svr,x1)
# plot the regression function
lines(x1, esvr.pred, type ="l", col = "blue", lwd =2)
```

Appendix 1.5

```
library(kernlab)

x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)
eps = 0.3
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
for(i in 1:length(x)){
arrows(x[i],y[i]-eps,x[i],y[i]+eps,length=0,code=3,angle=90)
}
# fit the epsilon support vector regression
eps.svr <- ksvm(x,y, scaled = FALSE, kernel="rbfdot", kpar=list(sigma=0.7), C=1,epsilon = eps)
# make predictions
x1<- seq(0,5,0.01)
esvr.pred <- predict(eps.svr,x1)
# plot the regression function
lines(x1, esvr.pred, type ="l", col = "blue", lwd =2)
```

Appendix 1.6

```
library(kernlab)

x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)
eps = 0.4
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
for(i in 1:length(x)){
arrows(x[i],y[i]-eps,x[i],y[i]+eps,length=0,code=3,angle=90)
}
eps.svr <- ksvm(x,y, scaled = FALSE, kernel="rbfdot", kpar=list(sigma=0.05), C=1,epsilon = eps)
# make predictions
x1<- seq(0,5,0.01)
esvr.pred <- predict(eps.svr,x1)
# plot the regression function
lines(x1, esvr.pred, type ="l", col = "blue", lwd =2)
```

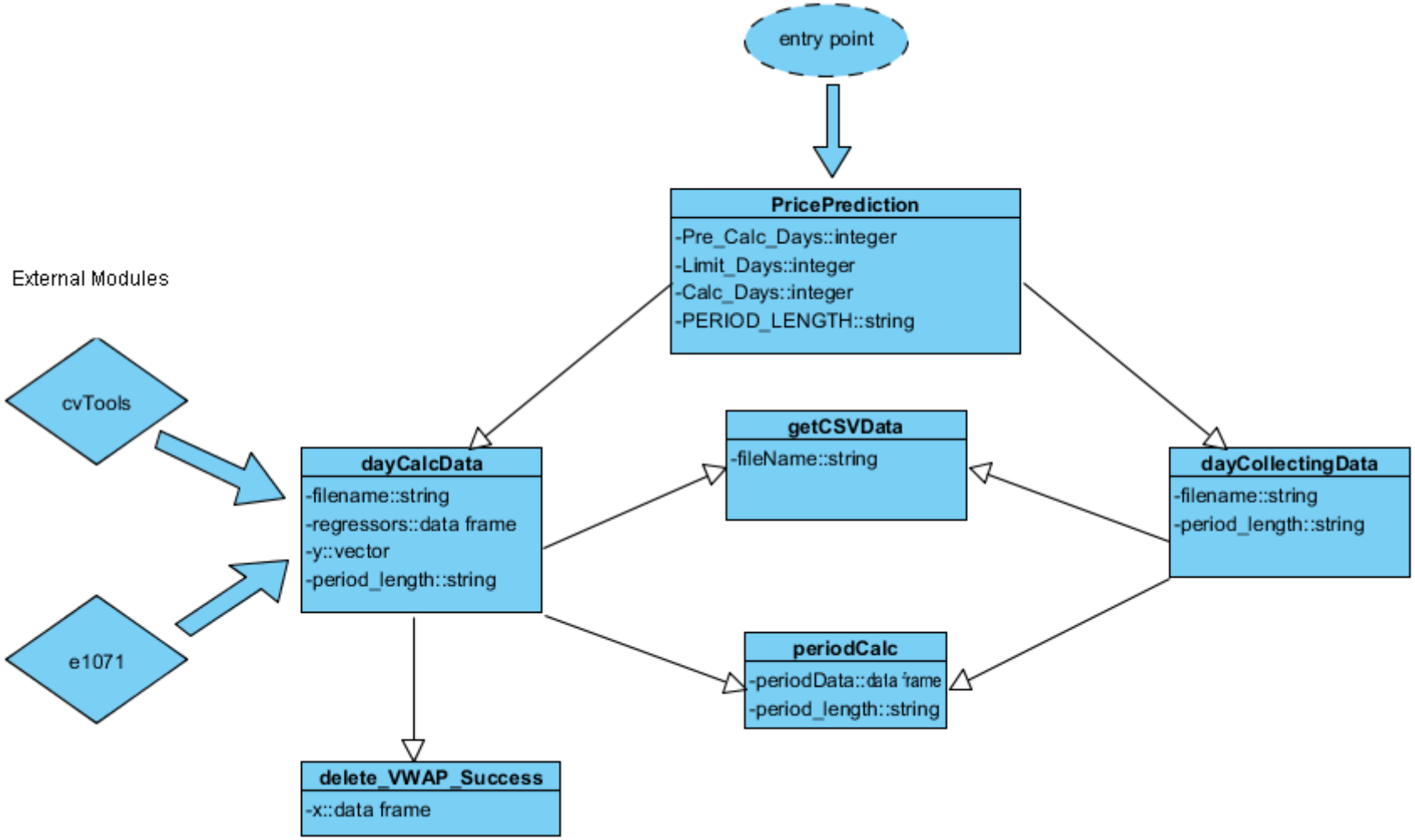
Appendix 1.7

```
library(kernlab)

x <- seq(0.5, 4.5, by = 1)
y <- c(1,2,1.5,3,2.5)

eps = 0.2
plot(x,y,ylim=c(0,4),xlim=c(0,5),pch=21,bg="red")
for(i in 1:length(x)){
arrows(x[i],y[i]-eps,x[i],y[i]+eps,length=0,code=3,angle=90)
}
eps.svr <- ksvm(x,y, scaled = FALSE, kernel="rbfdot", kpar=list(sigma=60), C=1,epsilon = eps)
#  make predictions
x1<- seq(0,5,0.01)
esvr.pred <- predict(eps.svr,x1)
# plot the regression function
lines(x1, esvr.pred, type ="l", col = "blue", lwd =2)
```

Appendix 2

date	sym	localtime	bid1	bid2	bid3	ask1	ask2	ask3	bsize1	bsize2	bsize3	asize1	asize2	asize3	rdersb	rdersb	rdersb	rdersa	rdersa	rdersa	_a_trad	price	size	tradetime
15.01.2007	AZN.L	08:04:34.752	2886	2885	2882	2888	2893	2899	3162	700	10000	1510	10000	12500	4	1	1	2	1	2	0	2886	1244	08:04:34.752
15.01.2007	AZN.L	08:04:34.922	2886	2885	2882	2888	2892	2893	3162	700	10000	1510	5000	10000	4	1	1	2	1	1	0	2886	1244	08:04:34.922
15.01.2007	AZN.L	08:04:35.362	2886	2885	2882	2888	2892	2893	3162	700	15000	1510	5000	10000	4	1	2	2	1	1	0	2886	1244	08:04:35.362
15.01.2007	AZN.L	08:04:35.572	2886	2885	2882	2888	2893	2899	3162	700	15000	1510	10000	12500	4	1	2	2	1	2	0	2886	1244	08:04:35.572
15.01.2007	AZN.L	08:04:36.532	2886	2885	2882	2888	2890	2893	3162	700	15000	1510	5000	10000	4	1	2	2	1	1	0	2886	1244	08:04:36.532
15.01.2007	AZN.L	08:04:37.842	2886	2885	2882	2888	2893	2899	3162	700	15000	1510	10000	12500	4	1	2	2	1	2	0	2886	1244	08:04:37.842
15.01.2007	AZN.L	08:04:38.172	2886	2885	2882	2888	2899	2900	3162	700	15000	1510	12500	1000	4	1	2	2	2	1	0	2886	1244	08:04:38.172
15.01.2007	AZN.L	08:04:38.172	2886	2885	2882	2888	2892	2899	3162	700	15000	1510	10000	12500	4	1	2	2	1	2	0	2886	1244	08:04:38.172
15.01.2007	AZN.L	08:04:38.302	2886	2885	2882	2888	2892	2899	3162	700	15000	1510	15000	12500	4	1	2	2	2	2	0	2886	1244	08:04:38.302
15.01.2007	AZN.L	08:04:38.302	2886	2885	2882	2888	2892	2899	3162	700	15000	1113	15000	12500	4	1	2	1	2	2	1	2888	-397	08:04:38.302
15.01.2007	AZN.L	08:04:38.312	2886	2885	2882	2888	2892	2899	3162	700	15000	0	15000	12500	4	1	2	2	2	1	1	2888	-1113	08:04:38.312
15.01.2007	AZN.L	08:04:38.312	2886	2885	2882	2892	2899	2900	3162	700	15000	15000	12500	1000	4	1	2	2	2	1	0	2888	-1113	08:04:38.312
15.01.2007	AZN.L	08:04:38.352	2888	2886	2885	2892	2899	2900	637	3162	700	15000	12500	1000	1	4	1	2	2	1	0	2888	-1113	08:04:38.352
15.01.2007	AZN.L	08:04:38.462	2888	2886	2885	2892	2899	2900	2147	3162	700	15000	12500	1000	2	4	1	2	2	1	0	2888	-1113	08:04:38.462
15.01.2007	AZN.L	08:04:38.522	2888	2886	2885	2892	2899	2900	2147	3162	700	15760	12500	1000	2	4	1	3	2	1	0	2888	-1113	08:04:38.522
15.01.2007	AZN.L	08:04:40.481	2888	2886	2885	2892	2899	2900	637	3162	700	15760	12500	1000	1	4	1	3	2	1	0	2888	-1113	08:04:40.481
15.01.2007	AZN.L	08:04:41.621	2888	2886	2885	2892	2899	2900	637	3107	700	15760	12500	1000	1	3	1	3	2	1	0	2888	-1113	08:04:41.621
15.01.2007	AZN.L	08:04:41.621	2888	2886	2885	2892	2899	2900	692	3107	700	15760	12500	1000	2	3	1	3	2	1	0	2888	-1113	08:04:41.621
15.01.2007	AZN.L	08:04:41.621	2888	2886	2885	2892	2899	2900	55	3107	700	15760	12500	1000	1	3	1	3	2	1	0	2888	-1113	08:04:41.621
15.01.2007	AZN.L	08:04:42.371	2888	2886	2885	2892	2899	2900	55	2884	700	15760	12500	1000	1	2	1	3	2	1	0	2888	-1113	08:04:42.371
15.01.2007	AZN.L	08:04:42.371	2889	2888	2886	2892	2899	2900	223	55	2884	15760	12500	1000	1	1	2	3	2	1	0	2888	-1113	08:04:42.371
15.01.2007	AZN.L	08:04:44.451	2889	2888	2886	2892	2899	2900	223	55	2884	16531	12500	1000	1	1	2	4	2	1	0	2888	-1113	08:04:44.451
15.01.2007	AZN.L	08:04:44.851	2889	2886	2885	2892	2899	2900	223	2884	700	16531	12500	1000	1	2	1	4	2	1	0	2888	-1113	08:04:44.851
15.01.2007	AZN.L	08:04:44.851	2889	2886	2885	2892	2899	2900	278	2884	700	16531	12500	1000	2	2	1	4	2	1	0	2888	-1113	08:04:44.851
15.01.2007	AZN.L	08:04:44.991	2889	2886	2885	2892	2899	2900	278	2884	700	16959	12500	1000	2	2	1	5	2	1	0	2888	-1113	08:04:44.991
15.01.2007	AZN.L	08:04:45.551	2889	2886	2885	2892	2899	2900	278	2884	700	16959	12500	1000	2	2	1	5	2	1	0	2888	-1113	08:04:45.551
15.01.2007	AZN.L	08:04:45.611	2889	2886	2885	2892	2899	2900	278	2884	700	16959	12500	1000	2	2	1	5	2	1	0	2888	-1113	08:04:45.611
15.01.2007	AZN.L	08:04:45.931	2889	2886	2885	2892	2899	2900	55	2884	700	16959	12500	1000	1	2	1	5	2	1	1	2889	223	08:04:45.931
15.01.2007	AZN.L	08:04:45.931	2889	2886	2885	2892	2899	2900	0	2884	700	16959	12500	1000	2	1	1	5	2	1	1	2889	55	08:04:45.931
15.01.2007	AZN.L	08:04:45.931	2886	2885	2882	2892	2899	2900	2884	700	10000	16959	12500	1000	2	1	1	5	2	1	0	2889	55	08:04:45.931
15.01.2007	AZN.L	08:04:47.121	2886	2885	2882	2892	2899	2900	2884	700	10000	11959	12500	1000	2	1	1	4	2	1	0	2889	55	08:04:47.121
15.01.2007	AZN.L	08:04:47.421	2886	2885	2882	2891	2892	2899	2884	700	10000	4300	11959	12500	2	1	1	1	4	2	0	2889	55	08:04:47.421

Appendix 3


entry point
PricePrediction
-Pre_Calc_Days::integer
-Limit_Days::integer
-Calc_Days::integer
-PERIOD_LENGTH::string
External Modules
cvTools
e1071
dayCalcData
-filename::string
-regressors::data frame
-y::vector
-period_length::string
getCSVData
-fileName::string
dayCollectingData
-filename::string
-period_length::string
periodCalc
-periodData::data frame
-period_length::string
delete_VWAP_Success
-x::data frame

Appendix 4

Bibliography

[1] **Orlov, A.** (2012). *Development of trading algorithms for taking or clearing a large single company stock position in a one day frame based on the historical tick data.* Master's thesis. University of Tartu, Faculty of mathematics and computer science, Institute of mathematical statistics

[2] Unsupervised learning. [WWW]

http://en.wikipedia.org/wiki/Unsupervised_learning (15.05.2013)

[3] Statistical classification. [WWW]

http://en.wikipedia.org/wiki/Statistical_classification (15.05.2013)

[4] Empirical risk minimization. [WWW]

http://en.wikipedia.org/wiki/Empirical_risk_minimization (15.05.2013)

[5] Schölkopf, B., Smola, A. (2002). *Learning with Kernels*, The MIT Press

[6] Kaasik, Ü., Kivistik, L. (1982). *Operatsioonianalüüs*. Valgus

[7] Support Vector Machines in R. [WWW]

http://www.jstatsoft.org/v15/i09/paper (17.05.2013)

[8] Cross-Validation. [WWW]

http://www.cse.iitb.ac.in/~tarung/smt/papers_ppt/ency-cross-validation.pdf (17.05.2013)

[9] Cross-validation(statistics). [WWW]

http://en.wikipedia.org/wiki/Cross-validation_(statistics) (17.05.2013)

[10] Order(exchange). [WWW]

http://en.wikipedia.org/wiki/Order_(exchange) (18.05.2013)

[11] **Aldridge, I.** (2010). *High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems*, John Wiley & Sons, Inc.

[12] **Kangro, R.** (2012). *Aegridade analüüs*. Loengumaterjalid, Tartu Ülikool

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Vassili Mušnikov (15.05.1987)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

„On the usage of support vector machines for the short-time price movement prediction in intra-day trading"

mille juhendaja on Raul Kangro.

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 20.05.2013