Order Execution Strategies With Support Vector Machines

Marcin Orchel

Abstract

In this report, we propose an application of support vector regression (SVR) for executing orders on stock markets. We use SVR for predicting a function of volume participation. We propose the improvement of predicting participation function by using support vector machines (SVM) with incorporated additional nonlinear constraint to the problem. We show that quality of the prediction influences execution costs. Moreover, we show how we can incorporate knowledge about stock prices. We compared ε-insensitive support vector regression (ε-SVR) and δ support vector regression (δ-SVR) with simple predictors such as the average price of execution from previous days. The tests were performed on data for stocks from NASDAQ-100 index. For both methods we achieved smaller variance of execution costs. Moreover, we decreased costs of order execution by using prediction of stock prices.

Contents

Chapter 1

Introduction

1.1 Support Vector Classification Basics

For a classification problem, we consider a set of n training vectors $\vec{x_i}$ for $i \in \{1, \ldots, n\}$, where $\vec{x_i} = \left(x_i^1, \ldots, x_i^m\right)$. The i-th training vector is mapped to $y_{\mathrm{c}}^i \in \{-1, 1\}$. The m is a dimension of the problem. The support vector classification (SVC) optimization problem for hard margin case with $\|\cdot\|_1$ norm is

OP 1.

$$\min_{\vec{w_{\mathrm{c}}}, b_{\mathrm{c}}} \quad f\left(\vec{w_{\mathrm{c}}}, b_{\mathrm{c}}\right) = \|\vec{w_{\mathrm{c}}}\|^2 \tag{1.1}$$

subject to

$$y_{\mathrm{c}}^i h\left(\vec{x_i}\right) \geq 1 \tag{1.2}$$

for $i \in \{1, \ldots, n\}$, where

$$h\left(\vec{x_i}\right) = \vec{w_{\mathrm{c}}} \cdot \vec{x_i} + b_{\mathrm{c}} \quad . \tag{1.3}$$

All points must be correctly classified Fig. 1.1a. The SVC soft margin case optimization problem with $\|\cdot\|_1$ norm is

OP 2.

$$\min_{\vec{w_{\mathrm{c}}}, b_{\mathrm{c}}, \vec{\xi_{\mathrm{c}}}} \quad f\left(\vec{w_{\mathrm{c}}}, b_{\mathrm{c}}, \vec{\xi_{\mathrm{c}}}\right) = \frac{1}{2}\|\vec{w_{\mathrm{c}}}\|^2 + C_{\mathrm{c}} \sum_{i=1}^{n} \xi_{\mathrm{c}}^i \tag{1.4}$$

subject to

$$y_{\mathrm{c}}^i h\left(\vec{x_i}\right) \geq 1 - \xi_{\mathrm{c}}^i \tag{1.5}$$

$$\vec{\xi_{\mathrm{c}}} \geq 0 \tag{1.6}$$

for $i \in \{1, \ldots, n\}$, where

$$h\left(\vec{x_i}\right) = \vec{w_{\mathrm{c}}} \cdot \vec{x_i} + b_{\mathrm{c}} \quad , \tag{1.7}$$

$$C_c > 0 \quad . \tag{1.8}$$

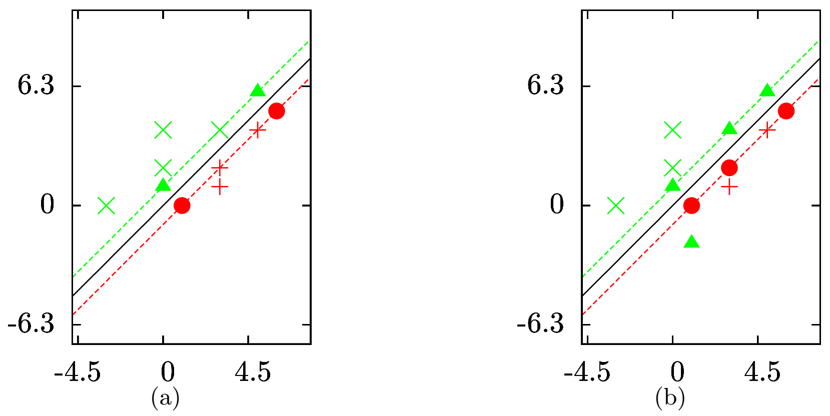

Figure 1.1: Two types of margin classifiers: hard and soft. Example points, support vectors (triangles and circles), solutions (solid lines), margin lines (dashed lines). (a) Hard. (b) Soft. A misclassified point is in (1, -2)

The $h^* (\vec{x}) = \vec{w_c^*} \cdot \vec{x} + b_c^* = 0$ is a decision curve of the classification problem. Some of training points can be incorrectly classified Fig. 1.1b.

The $\vec{w_c^*} \cdot \vec{x}$ can be computed as

$$\vec{w_c^*} \cdot \vec{x} = \sum_{i=1}^{n} y_c^i \alpha_i^* K (\vec{x_i}, \vec{x}) \ . \tag{1.9}$$

Therefore, the decision curve is

$$h^* (\vec{x}) = \sum_{i=1}^{n} y_c^i \alpha_i^* K (\vec{x_i}, \vec{x}) + b_c^* = 0 \ , \tag{1.10}$$

where α_i are Lagrange multipliers of the dual problem, $K (\cdot, \cdot)$ is a kernel function, which appears only in the dual problem. *Margin boundaries* are defined as the two hyperplanes $h (\vec{x}) = -1$ and $h (\vec{x}) = 1$. *Optimal margin boundaries* are defined as the two hyperplanes $h^* (\vec{x}) = -1$ and $h^* (\vec{x}) = 1$. *Geometric margin of the hyperplane* h is defined as $1/ \|\vec{w_c}\|$. The i-th training example is *a support vector*, when $\alpha_i^* \neq 0$. A set of support vectors contains all training examples lying below optimal margin boundaries ($y_c^i h^* (\vec{x_i}) < 1$), and part of the examples lying exactly on the optimal margin boundaries ($y_c^i h^* (\vec{x_i}) = 1$), Fig. 1.1b.

SVC Without the Offset

Another variant of SVC is the SVC without the offset b_c, analyzed recently in [10]. The optimization problem is the same except missing b_c term, for the soft case it is

OP 3.

$$\min_{\vec{w_c}, \vec{\xi_c}} \ f \left(\vec{w_c}, \vec{\xi_c} \right) = \frac{1}{2} \|\vec{w_c}\|^2 + C_c \cdot \vec{\xi_c} \tag{1.11}$$

subject to

$$y_{\mathrm{c}}^{i} h\left(\vec{x_i}\right) \geq 1-\xi_{\mathrm{c}}^{i} \tag{1.12}$$

$$\vec{\xi} \geq 0 \tag{1.13}$$

for $i \in \{1, \ldots, n\}$, where

$$h\left(\vec{x_i}\right)=\vec{w_{\mathrm{c}}} \cdot \vec{x_i} \ \ , \tag{1.14}$$

$$C_{\mathrm{c}}>0 \ \ . \tag{1.15}$$

The dual problem is

OP 4.

$$\max_{\vec{\alpha}} \quad d\left(\vec{\alpha}\right)=\vec{\alpha}-\frac{1}{2} \vec{\alpha}^{T} Q \vec{\alpha} \tag{1.16}$$

subject to

$$0 \leq \vec{\alpha} \leq C_{c} \tag{1.17}$$

where $Q_{ij}=y_i y_j K\left(\vec{x_i}, \vec{x_j}\right)$, for all $i, j \in \{1, \ldots, n\}$.

We can notice missing linear constraint. The decision curve is

$$h^{*}\left(\vec{x}\right)=\sum_{i=1}^{n} y_{\mathrm{c}}^{i} \alpha_{i}^{*} K\left(\vec{x_i}, \vec{x}\right)=0 \ \ . \tag{1.18}$$

1.2 Support Vector Regression Basics

In a regression problem, we consider a set of training vectors $\vec{x_i}$ for $i \in \{1, \ldots, n\}$, where $\vec{x_i}=\left(x_i^1, \ldots, x_i^m\right)$. The i-th training vector is mapped to $y_{\mathrm{r}}^{i} \in \mathbb{R}$. The m is a dimension of the problem. The ε-SVR soft case optimization problem is

OP 5.

$$\min_{\vec{w_{\mathrm{r}}}, b_{\mathrm{r}}, \vec{\xi_{\mathrm{r}}}, \vec{\xi_{\mathrm{r}}^{*}}} \quad f\left(\vec{w_{\mathrm{r}}}, b_{\mathrm{r}}, \vec{\xi_{\mathrm{r}}}, \vec{\xi_{\mathrm{r}}^{*}}\right)=\frac{1}{2}\left\|\vec{w_{\mathrm{r}}}\right\|^{2}+C_{\mathrm{r}} \sum_{i=1}^{n}\left(\xi_{\mathrm{r}}^{i}+\xi_{\mathrm{r}}^{*i}\right) \tag{1.19}$$

subject to

$$y_{\mathrm{r}}^{i}-g\left(\vec{x_i}\right) \leq \varepsilon+\xi_{\mathrm{r}}^{i} \tag{1.20}$$

$$g\left(\vec{x_i}\right)-y_{\mathrm{r}}^{i} \leq \varepsilon+\xi_{\mathrm{r}}^{i*} \tag{1.21}$$

$$\vec{\xi_{\mathrm{r}}} \geq 0 \tag{1.22}$$

$$\vec{\xi_{\mathrm{r}}^{*}} \geq 0 \tag{1.23}$$

for $i \in \{1, \ldots, n\}$, where

$$g\left(\vec{x_i}\right)=\vec{w_{\mathrm{r}}} \cdot \vec{x_i}+b_{\mathrm{r}} \ \ , \tag{1.24}$$

$$\varepsilon \in \mathbb{R} \ \ . \tag{1.25}$$

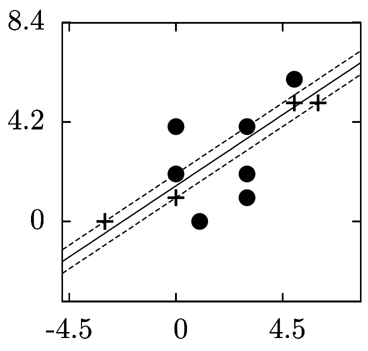

Figure 1.2: The idea of ε-SVR. Points - examples, circles - support vectors, a solid line - a solution, dashed lines - ε boundaries

The $g^* (\vec{x}) = \vec{w_r^*} \cdot \vec{x} + b_r^*$ is a regression function. Optimization problem 5 is transformed into an equivalent dual problem. The regression function becomes

$$g^* (\vec{x}) = \sum_{i=1}^{n} (\alpha_i^* - \beta_i^*) K (\vec{x_i}, \vec{x}) + b_r^* \ , \tag{1.26}$$

where α_i, β_i are Lagrange multipliers, $K (\cdot, \cdot)$ is a kernel function. The ε boundaries are defined as $g (\vec{x}) - \varepsilon$ and $g (\vec{x}) + \varepsilon$. The i-th training example is *a support vector*, when $\alpha_i^* - \beta_i^* \neq 0$. For $\varepsilon \geq 0$, a set of support vectors contains all training examples lying outside ε boundaries, and part of the examples lying exactly on ε boundaries, Fig. 1.2. The number of support vectors can be controlled by ε parameter.

1.3 SVM with C_i Weights

A 1-norm soft margin SVC optimization problem for training examples $\vec{x_i}$ with weights C_i is

OP 6.

$$\min_{\vec{w}, b, \vec{\xi}} \ f \left(\vec{w}, b, \vec{\xi} \right) = \frac{1}{2} \|\vec{w}\|^2 + \vec{C_c} \cdot \vec{\xi} \tag{1.27}$$

subject to

$$y_i h (\vec{x_i}) \geq 1 - \xi_i \tag{1.28}$$

$$\vec{\xi} \geq 0 \tag{1.29}$$

for $i \in \{1, \ldots, n\}$, where

$$\vec{C_c} \gg 0 \tag{1.30}$$

$$h (\vec{x_i}) = \vec{w} \cdot \vec{x_i} + b \ . \tag{1.31}$$

The C_i weights were also used with ε-SVR for predicting time series data, [11]. The other types of weights that were used with ε-SVR are ε_i weights per example replacing the parameter ε. They were used for density estimation, [12]. Sometimes we use different ε weights for inequalities (1.20), (1.21), ε_u and ε_d respectively. And finally we can combine above changes and use ε_u^i and ε_d^i weights

OP 7.

$$\min_{\vec{w_{\mathrm{r}}}, b_{\mathrm{r}}, \vec{\xi_{\mathrm{r}}}, \vec{\xi_{\mathrm{r}}^*}} f\left(\vec{w_{\mathrm{r}}}, b_{\mathrm{r}}, \vec{\xi_{\mathrm{r}}}, \vec{\xi_{\mathrm{r}}^*}\right) = \frac{1}{2}\left\|\vec{w_{\mathrm{r}}}\right\|^2 + \vec{C_{\mathrm{r}}}\sum_{i=1}^{n}\left(\xi_{\mathrm{r}}^i + \xi_{\mathrm{r}}^{*i}\right) \tag{1.32}$$

subject to

$$y_{\mathrm{r}}^i - g\left(\vec{x_i}\right) \le \varepsilon_u^i + \xi_{\mathrm{r}}^i \tag{1.33}$$

$$g\left(\vec{x_i}\right) - y_{\mathrm{r}}^i \le \varepsilon_d^i + \xi_{\mathrm{r}}^{i*} \tag{1.34}$$

$$\vec{\xi_{\mathrm{r}}} \ge 0 \tag{1.35}$$

$$\vec{\xi_{\mathrm{r}}^*} \ge 0 \tag{1.36}$$

for $i \in \{1, \ldots, n\}$, where

$$g\left(\vec{x_i}\right) = \vec{w_{\mathrm{r}}} \cdot \vec{x_i} + b_{\mathrm{r}} \quad . \tag{1.37}$$

We can notice that changing y_r^i value by Δy_r^i is equivalent to changing ε_u^i by $-\Delta y_r^i$ and changing ε_d^i by Δy_r^i.

1.4 Introduction to δ-SVR

We consider a set of training vectors $\vec{x_i}$ for $i \in \{1, \ldots, n\}$, where $\vec{x_i} = \left(x_i^1, \ldots, x_i^m\right)$. The i-th training vector is mapped to $y_{\mathrm{r}}^i \in \mathbb{R}$. The δ-SVR method finds a regression function by the following procedure.

1. Every training example $\vec{x_i}$ is duplicated, an output value y_{r}^i is increased by a value of a parameter $\delta \ge 0$ for original training examples, and decreased by δ for duplicated training examples.

2. Every training example $\vec{x_i}$ is converted to a classification example by incorporating the output to the input vector as an additional feature and setting class 1 for original training examples, class -1 for duplicated training examples.

3. The SVC method is launched for a classification problem.

4. The solution of SVC method is converted back to function form.

The result of the first step is a set of training mappings for $i \in \{1, \ldots, 2n\}$

$$\begin{cases} \vec{x_i} \to y_{\mathrm{r}}^i + \delta & \text{for} \quad i \in \{1, \ldots, n\} \\ \vec{x_i} \to y_{\mathrm{r}}^i - \delta & \text{for} \quad i \in \{n+1, \ldots, 2n\} \end{cases} \tag{1.38}$$

where $\vec{x_{n+i}} = \vec{x_i}$, $y_r^{n+i} = y_r^i$ for $i \in \{1, \ldots, n\}$, $\delta \in \mathbb{R}$. The δ is called *the translation parameter*. The result of the second step is a set of training mappings for $i \in \{1, \ldots, 2n\}$

$$\vec{c_i} = \left(x_i^1, \ldots, x_i^m, y_{\mathrm{r}}^i + y_c^i \delta\right) \rightarrow y_c^i \tag{1.39}$$

where $y_c^i = 1$ for $i \in \{1, \ldots, n\}$, and $y_c^i = -1$ for $i \in \{n+1, \ldots, 2n\}$. The dimension of the $\vec{c_i}$ vectors is equal to $m+1$. The set of $\vec{x_i}$ mappings before duplication is called *a regression data setting*, the set of $\vec{c_i}$ ones is called *a classification data setting*. In the third step, OP 2 is solved with $\vec{c_i}$ examples, so we can write OP 2 as

OP 8.

$$\min_{\vec{w_{\mathrm{c}}}, b_{\mathrm{c}}, \vec{\xi_{\mathrm{c}}}} \quad f\left(\vec{w_{\mathrm{c}}}, b_{\mathrm{c}}, \vec{\xi_{\mathrm{c}}}\right) = \frac{1}{2}\left\|\vec{w_{\mathrm{c}}}\right\|^2 + C_{\mathrm{c}} \sum_{i=1}^{2n} \xi_{\mathrm{c}}^i \tag{1.40}$$

subject to

$$y_{\mathrm{c}}^i \left(\vec{w_{\mathrm{c,red}}} \cdot \vec{x_i} + w_c^{m+1}\left(y_r^i + y_c^i \delta\right) + b_c\right) \geq 1 - \xi_{\mathrm{c}}^i \tag{1.41}$$

$$\vec{\xi_{\mathrm{c}}} \geq 0 \tag{1.42}$$

for $i \in \{1, \ldots, 2n\}$.

The $\vec{w_{\mathrm{c,red}}}$ is defined as $\vec{w_{\mathrm{c,red}}} = (w_1, \ldots, w_m)$. The $h^*\left(\vec{x}\right) = \vec{w_{\mathrm{c}}^*} \cdot \vec{x} + b_{\mathrm{c}}^* = 0$ is a decision curve of the classification problem. Note that $h^*\left(\vec{x}\right)$ is in the implicit form of the last coordinate of $\vec{x}$. In the fourth step, an explicit form of the last coordinate needs to be find. The explicit form is needed for example for testing new examples. The $\vec{w_{\mathrm{c}}}$ variable of the primal problem for a simple linear kernel is found in the following way

$$\vec{w_{\mathrm{c}}} = \sum_{i=1}^{2n} y_{\mathrm{c}}^i \alpha_i \vec{c_i} \quad . \tag{1.43}$$

For a simple linear kernel the explicit form of (1.10) is

$$x_{m+1} = \frac{-\sum_{j=1}^{m} w_{\mathrm{c}}^j x_j - b_{\mathrm{c}}}{w_{\mathrm{c}}^{m+1}} \quad . \tag{1.44}$$

The regression solution is $g^*\left(\vec{x}\right) = \vec{w_{\mathrm{r}}} \cdot \vec{x} + b_{\mathrm{r}}$, where $w_{\mathrm{r}}^i = -w_{\mathrm{c}}^i / w_{\mathrm{c}}^{m+1}$, $b_{\mathrm{r}} = -b_{\mathrm{c}} / w_{\mathrm{c}}^{m+1}$ for $i = 1, \ldots, m$. For nonlinear kernels, a conversion to the explicit form has some limitations. First, a decision curve could have more than one value of the last coordinate for specific values of remaining coordinates of $\vec{x}$ and therefore it cannot be converted unambiguously to the function (e.g. a polynomial kernel with a dimension equals to 2). Second, even when the conversion to the function is possible, there is no explicit analytical formula (e.g. a polynomial kernel with a dimension greater than 4), hence a special method for finding the explicit formula of the coordinate should be used, e.g. a bisection method. The disadvantage of this solution is a longer time of testing new examples. To overcome these problems, we proposed to incorporate prior knowledge to the classification problem, that the solution will be always in the form of the function in the chosen direction. Thus, we proposed in [7] a new kernel type in which the last coordinate is placed only inside a linear term. The new kernel is constructed from an original kernel by removing the last coordinate, and adding the linear term with the last coordinate

$$K\left(\vec{x}, \vec{y}\right) = K_o\left(\vec{x_{\text{red}}}, \vec{y_{\text{red}}}\right) + x_{m+1} y_{m+1} \quad , \tag{1.45}$$

where $\vec{x}$ and $\vec{y}$ are $m+1$ dimensional vectors, $\vec{x_{\text{red}}} = (x_1, \ldots, x_m)$, $\vec{y_{\text{red}}} = (y_1, \ldots, y_m)$, $K_{\text{o}}\left(\cdot, \cdot\right)$ is the original kernel from which the new one was constructed. For the most popular kernels polynomial, radial basis function (RBF) and sigmoid, the conversions are respectively

$$\left(\vec{x} \cdot \vec{y}\right)^d \quad \rightarrow \quad \left(\sum_{i=1}^{m} x_i y_i\right)^d + x_{m+1} y_{m+1} \quad , \tag{1.46}$$

$$\exp - \frac{\left\|\vec{x} - \vec{y}\right\|^2}{2\sigma^2} \quad \rightarrow \quad \exp - \frac{\sum_{i=1}^{m} \left(x_i - y_i\right)^2}{2\sigma^2} + x_{m+1} y_{m+1} \quad , \tag{1.47}$$

$$\tanh \vec{x}\vec{y} \quad \rightarrow \quad \tanh \sum_{i=1}^{m} x_i y_i + x_{m+1} y_{m+1} \quad , \tag{1.48}$$

where $\vec{x}$ and $\vec{y}$ are $m+1$ dimensional vectors. For the new kernel type, the explicit form of (1.10) for δ-SVR is

$$x_{m+1} = \frac{- \sum_{i=1}^{2n} y_{\text{c}}^i \alpha_i K_{\text{o}}\left(\vec{x_i}, \vec{x_{\text{red}}}\right) - b_{\text{c}}}{\sum_{i=1}^{2n} y_{\text{c}}^i \alpha_i c_i^{m+1}} \quad . \tag{1.49}$$

1.4.1 Weighting the Translation Parameter

We can consider incorporating prior knowledge by setting different values of the translation parameter for each example, so we can have δ_i parameters, for $i \in \{1, \ldots, n\}$ and the same parameters for $i \in \{n+1, \ldots, 2n\}$. We can also consider setting different values of δ for up and down translations, so we can have two parameters: δ_{u} and δ_{d}. And finally we can also consider the parameters δ_{u}^i and δ_{d}^i (additionally with the $\vec{C_c}$ weight)

OP 9.

$$\min_{\vec{w_{\text{c}}}, b_{\text{c}}, \vec{\xi_{\text{c}}}} f\left(\vec{w_{\text{c}}}, b_{\text{c}}, \vec{\xi_{\text{c}}}\right) = \frac{1}{2}\left\|\vec{w_{\text{c}}}\right\|^2 + \vec{C_{\text{c}}} \sum_{i=1}^{2n} \xi_{\text{c}}^i \tag{1.50}$$

subject to

$$y_{\text{c}}^i \left(\vec{w_{\text{c,red}}} \cdot \vec{x_i} + w_c^{m+1}\left(y_r^i + y_c^i \delta_i\right) + b_c\right) \geq 1 - \xi_{\text{c}}^i \tag{1.51}$$

$$\vec{\xi_{\text{c}}} \geq 0 \tag{1.52}$$

for $i \in \{1, \ldots, 2n\}$, where $\delta_i = \delta_u^i$ for $i \in \{1, \ldots, n\}$ and $\delta_i = \delta_d^i$ for $i \in \{n+1, \ldots, 2n\}$.

We define the margin of an example (sometimes called just a margin) in the following way:

Definition 1.4.1. Given some curve $h(\vec{x}) = 0$, *the margin of the $\vec{x_p}$ example* is defined as a value $|h(\vec{x_p})|$.

For hard margin SVC, OP 1, the margin of the closest examples is equal to 1. *The knowledge about the margin of an example* (sometimes called the knowledge about a margin) is defined as prior information about the margins of particular examples.

1.5 Introduction to φ-SVC

The φ support vector classification (φ-SVC) method is a recently proposed method of incorporating knowledge about the margin of an example to SVC, [5, 6, 8]. The φ-SVC optimization problem is defined with an additional parameter per example added in the right side of the inequality (1.5). Another modification of inequality constraints was proposed in [13]. The authors modify the inequalities by multiplying the left side of the inequalities by some monotonically decreasing function of additional example weights. The φ-SVC is a more general concept of weights per example with any values possible and with different interpretation.

Now, we will closely look at φ-SVC optimization problem. We define φ-SVC optimization problem based on SVC with cost weights per example OP 6 as

OP 10.

$$\min_{\vec{w_c}, b_c, \vec{\xi_c}} \quad f\left(\vec{w_c}, b_c, \vec{\xi_c}\right) = \frac{1}{2}\|\vec{w_c}\|^2 + \vec{C_c} \cdot \vec{\xi_c} \tag{1.53}$$

subject to

$$y_c^i h(\vec{x_i}) \geq 1 + \varphi_i - \xi_c^i \tag{1.54}$$

$$\vec{\xi_c} \geq 0 \tag{1.55}$$

for $i \in \{1, \dots, n\}$, where

$$\vec{C_c} \gg 0 \tag{1.56}$$

$$\varphi_i \in \mathbb{R} \tag{1.57}$$

$$h(\vec{x_i}) = \vec{w_c} \cdot \vec{x_i} + b_c \quad . \tag{1.58}$$

The new weights φ_i are present only in (1.54). When $\vec{\varphi} = 0$, the OP 10 is equivalent to OP 6. When all φ_i are equal to some constant $\varphi > -1$, we will get the same decision boundary as for $\varphi = 0$ when we change $\vec{C_c}$ to $\vec{C_c}/(1+\varphi)$.

Proof. We will prove that we get the same decision boundary when we replace 1 with $1/d$, for some $d > 0$, $\varphi_i = 0$. Let's replace ξ_i with ξ_i/d and we get the inequalities $y_c^i h(\vec{x_i}) \geq 1/d - \xi_c^i/d$. After multiplying by d we get $y_c^i dh(\vec{x_i}) \geq 1 - \xi_c^i$. The objective function can be multiplied by d^2 and we get $\frac{1}{2}\left\|d\vec{w_c}\right\|^2 + d\vec{C_c} \cdot \vec{\xi_c}$, the inequalities $\xi_i/d \geq 0$ can be replaced by $\xi_i \geq 0$. So we get the same optimization problem as the original one with the new $\vec{C_c} = d\vec{C_c}$ and with the new decision curve $dh(\vec{x}) = 0$. □

We can notice that when neglecting ξ_{c}^i, we get different bounds for the margin: when $y_{\mathrm{c}}^i = 1$ and $g(\vec{x}_i) \geq 0$, then we get a lower bound $1 + \varphi_i$, when $y_{\mathrm{c}}^i = -1$ and $g(\vec{x}_i) \geq 0$, then we get an upper bound $-(1 + \varphi_i)$, when $y_{\mathrm{c}}^i = 1$ and $g(\vec{x}_i) < 0$, then we get an upper bound $-(1 + \varphi_i)$, when $y_{\mathrm{c}}^i = -1$ and $g(\vec{x}_i) < 0$, then we get a lower bound $1 + \varphi_i$. We can distinguish three cases: $1 + \varphi_i > 0$, $1 + \varphi_i < 0$ and $1 + \varphi_i = 0$. For the first one, we get a lower bound on the margin equal to $1 + \varphi_i$. For the second, we get an upper bound on the margin equal to $-(1 + \varphi_i)$, for the third, we get an upper bound on the margin equal to 0. Therefore, by using φ_i weights, we can incorporate knowledge about the margin of an example.

The next property of φ_i weights is that they have impact on the distance between the curve $h(\vec{x}) = 0$ and the i-th example. We can conduct the same analysis as above for distances by dividing both sides of (1.54) by $\|\vec{w}_{\mathrm{c}}\|$, so we get $y_{\mathrm{c}}^i h(\vec{x}_i) / \|\vec{w}_{\mathrm{c}}\| \geq 1/\|\vec{w}_{\mathrm{c}}\| + \varphi_i/\|\vec{w}_{\mathrm{c}}\|$. The conclusion is similar: for the case when $1 + \varphi_i > 0$, we get a lower bound on the distance equal to $(1 + \varphi_i)/\|\vec{w}_{\mathrm{c}}\|$. For the case when $1 + \varphi_i < 0$, we get an upper bound on the distance equal to $-(1 + \varphi_i)/\|\vec{w}_{\mathrm{c}}\|$. For the case when $1 + \varphi_i = 0$, we get an upper bound on the distance equal to 0.

Note that when we take into account ξ_{c}^i, we can see that knowledge about the margin of an example is incorporated as imperfect prior knowledge. The violation of knowledge about the margin of an example is controlled by the C_{c}^i parameters. Comparing loosely φ_i weights with slack variables: φ_i weights are constant, they are absent in the objective function, whereas a sum of slack variables is minimized as part of the objective function.

We can also derive the equivalent optimization problem to OP 10, where φ_i weights are present in the constraints with slack variables

OP 11.

$$\min_{\vec{w}_{\mathrm{c}}, b_{\mathrm{c}}, \vec{\xi}_{\mathrm{c}}} f\left(\vec{w}_{\mathrm{c}}, b_{\mathrm{c}}, \vec{\xi}_{\mathrm{c}}\right) = \frac{1}{2}\|\vec{w}_{\mathrm{c}}\|^2 + \vec{C}_{\mathrm{c}} \cdot \vec{\xi}_{\mathrm{c}} \tag{1.59}$$

subject to

$$y_{\mathrm{c}}^i h(\vec{x}_i) \geq 1 - \xi_{\mathrm{c}}^i \tag{1.60}$$

$$\vec{\xi} \geq \varphi_i \tag{1.61}$$

for $i \in \{1, \ldots, n\}$.

1.6 Incorporating Linear Dependency of Function Values

Another example of application for knowledge about the margin of an example is the incorporation of the additional constraint in the form of a constant value of linear dependency on function values, that the solution must satisfy, for regression it is

$$\sum_{i=1}^{s} s_i g\left(\vec{d}_i\right) = e \ , \tag{1.62}$$

where s_i are some parameters, for which $\sum_{i=1}^{s} s_i \neq 0$, d_i are some points, s is the number of d_i points, e is a parameter, g is defined in (1.24), and for classification it is

$$\sum_{i=1}^{s} s_i h\left(\vec{d}_i\right) = e \quad . \tag{1.63}$$

where s_i are some parameters, for which $\sum_{i=1}^{s} s_i \neq 0$, d_i are some points, s is the number of d_i points, e is a parameter.

We will show how to incorporate this constraint to φ-SVC, δ-SVR, and ε-SVR. Knowledge about the margin of an example is used in incorporation of this constraint to φ-SVC and ε-SVR.

1.6.1 Incorporating Linear Dependency on Function Values to φ-SVC

Incorporating (1.63) to OP 10 leads to the φ-SVC optimization problem without the offset, which is the SVC optimization problem without the offset OP 3 with additional margin weights (and $\vec{C}_c$ weights for completeness)

OP 12.

$$\min_{\vec{w}_c, \vec{\xi}_c} \quad f\left(\vec{w}_c, \vec{\xi}_c\right) = \frac{1}{2}\left\|\vec{w}_c\right\|^2 + \vec{C}_c \cdot \vec{\xi}_c \tag{1.64}$$

subject to

$$y_c^i h\left(\vec{x}_i\right) \geq 1 - \xi_c^i + \varphi_i \tag{1.65}$$

$$\vec{\xi} \geq 0 \tag{1.66}$$

for $i \in \{1, \ldots, n\}$, where

$$\vec{C}_c \gg 0 \tag{1.67}$$

$$\varphi_c^i \in \mathbb{R} \tag{1.68}$$

$$h\left(\vec{x}_i\right) = \vec{w}_c \cdot \vec{x}_i \quad . \tag{1.69}$$

The dual problem compared to OP 4 contains additionally margin weights in an objective function

OP 13.

$$\max_{\vec{\alpha}} \quad d\left(\vec{\alpha}\right) = \vec{\alpha} \cdot \left(1 + \vec{\varphi}\right) - \frac{1}{2}\vec{\alpha}^T Q \vec{\alpha} \tag{1.70}$$

subject to

$$0 \leq \vec{\alpha} \leq \vec{C}_c \tag{1.71}$$

where $Q_{ij} = y_i y_j K\left(\vec{x}_i, \vec{x}_j\right)$, for all $i, j \in \{1, \ldots, n\}$.

Incorporating (1.63) to OP 10 also leads to the new kernel in the form of transformation of the input vectors

$$\vec{x} \to \vec{x} - \frac{1}{\sum_{i=1}^{s} s_i} \sum_{i=1}^{s} s_i \vec{d}_i \quad , \tag{1.72}$$

$$K\left(\vec{x},\vec{y}\right)=\left(\vec{x}-\frac{1}{\sum_{i=1}^{s}s_i}\sum_{i=1}^{s}s_i\vec{d_i}\right)\left(\vec{y}-\frac{1}{\sum_{i=1}^{s}s_i}\sum_{i=1}^{s}s_i\vec{d_i}\right)\ . \tag{1.73}$$

This is a symmetrical kernel. We set the margin weights as

$$\varphi_i=\varphi_{old}-y_i\frac{e}{\sum_{i=1}^{s}s_i}\ . \tag{1.74}$$

We propose also a nonlinear kernel by further kernelization of (1.73), we get

$$K\left(\vec{x},\vec{y}\right)=K_o\left(\vec{x},\vec{y}\right)-\frac{1}{\sum_{i=1}^{s}s_i}\sum_{i=1}^{s}s_iK_o\left(\vec{x},\vec{d_i}\right)-\frac{1}{\sum_{i=1}^{s}s_i}\sum_{i=1}^{s}s_iK_o\left(\vec{y},\vec{d_i}\right) \tag{1.75}$$

$$+\frac{1}{\left(\sum_{i=1}^{s}s_i\right)^2}\sum_{i=1}^{s}\sum_{j=1}^{s}s_is_jK_o\left(\vec{d_i},\vec{d_j}\right)\ . \tag{1.76}$$

This is also a symmetrical kernel. In the final solution, we use the transformation of the input vectors (1.72) only for the training vectors present in the term $\vec{w_c}$, therefore for the solution, we get the following kernel

$$K\left(\vec{x_j},\vec{x}\right)=K_o\left(\vec{x_j},\vec{x}\right)-\frac{1}{\sum_{i=1}^{s}s_i}\sum_{i=1}^{s}s_iK_o\left(\vec{d_i},\vec{x}\right)\ , \tag{1.77}$$

where $j\in\{1,\dots,n\}$. Note that this is a kernel between a training vector and $\vec{x}$ used only for defining the solution. For the final solution, we compute the offset as

$$b=\frac{1}{\sum_{i=1}^{s}s_i}\left(e-\sum_{i=1}^{s}s_i\sum_{j=1}^{n}\alpha_jy_c^jK_o\left(\vec{x_j},\vec{d_i}\right)\right) \tag{1.78}$$

$$+\frac{1}{\left(\sum_{i=1}^{s}s_i\right)^2}\sum_{i=1}^{s}s_i\sum_{j=1}^{n}\sum_{k=1}^{s}\alpha_jy_c^js_kK_o\left(\vec{d_k},\vec{d_i}\right)\ . \tag{1.79}$$

1.6.2 Incorporating the Linear Dependency on Function Values to ε-SVR

We incorporated (1.62) to OP 5. Because ε-SVR is a special case of φ-SVC, and we have derived already the incorporation for φ-SVC, for ε-SVR we set the following weights

$$\varphi_i=y_c^iy_r^i-\varepsilon-1-y_c^i\frac{e}{\sum_{i=1}^{s}s_i} \tag{1.80}$$

for $i\in\{1,\dots,2n\}$. We also use input space transformation. We use φ-SVC without the offset.

1.6.3 Incorporating Linear Dependency on Function Values to δ-SVR

For completeness we show here the incorporation of linear dependency on function values to δ-SVR. We do not use margin weights in this case. We use kernels developed for δ-SVR, (1.45). Incorporating (1.62) to δ-SVR leads to the SVC optimization problem without the offset OP 3 with the new kernel

$$K\left(\vec{x},\vec{y}\right) = \left(\vec{x_{\text{red}}} - \frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i \vec{d_{\text{i,red}}}\right)\left(\vec{y_{\text{red}}} - \frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i \vec{d_{\text{i,red}}}\right) \tag{1.81}$$

$$+\left(x_{m+1} - \frac{e}{\sum_{i=1}^{s} s_i}\right)\left(y_{m+1} - \frac{e}{\sum_{i=1}^{s} s_i}\right) \quad , \tag{1.82}$$

where $\vec{d_{i,\text{red}}} = \left(d_i^1,\ldots,d_i^m\right)$. This is a symmetrical kernel. We propose also a nonlinear kernel by further kernelization of (1.81), we get

$$K\left(\vec{x},\vec{y}\right) = K_o\left(\vec{x_{\text{red}}},\vec{y_{\text{red}}}\right) - \frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i K_o\left(\vec{d_{\text{i,red}}},\vec{x_{\text{red}}}\right) \tag{1.83}$$

$$-\frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i K_o\left(\vec{d_{\text{i,red}}},\vec{y_{\text{red}}}\right) + \frac{1}{\left(\sum_{i=1}^{s} s_i\right)^2}\sum_{i=1}^{s}\sum_{j=1}^{s} s_i s_j K_o\left(\vec{d_{\text{i,red}}},\vec{d_{\text{j,red}}}\right) \tag{1.84}$$

$$+x_{m+1}y_{m+1} - x_{m+1}\frac{e}{\sum_{i=1}^{s} s_i} - y_{m+1}\frac{e}{\sum_{i=1}^{s} s_i} + \frac{e^2}{\left(\sum_{i=1}^{s} s_i\right)^2} \quad . \tag{1.85}$$

This is also a symmetrical kernel. In the final solution, we use the kernelization process only for the training vectors present in the term $\vec{w_{\text{c}}}$, therefore for the solution, we get the following kernel

$$K\left(\vec{x_j},\vec{x}\right) = K_o\left(\vec{x_{\text{j,red}}},\vec{x_{\text{red}}}\right) - \frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i K_o\left(\vec{d_{\text{i,red}}},\vec{x_{\text{red}}}\right) + \left(x_j^{m+1} - \frac{e}{\sum_{i=1}^{s} s_i}\right)x_{m+1} \quad , \tag{1.86}$$

where $j \in \{1,\ldots,n\}$. Note that this is a kernel between a training vector and $\vec{x}$ used only for defining the solution. For the final solution, we compute the offset as

$$b_{\text{c}} = -\frac{1}{\sum_{i=1}^{s} s_i} e \sum_{j=1}^{2n}\alpha_j y_{\text{c}}^j\left(x_j^{m+1} - \frac{e}{\sum_{i=1}^{s} s_i}\right) - \frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i \sum_{j=1}^{2n}\alpha_j y_{\text{c}}^j K_o\left(\vec{x_{\text{j,red}}},\vec{d_{\text{i,red}}}\right) \tag{1.87}$$

$$+\frac{1}{\sum_{i=1}^{s} s_i}\sum_{i=1}^{s} s_i \sum_{j=1}^{2n}\alpha_j y_{\text{c}}^j \frac{1}{\sum_{i=1}^{s} s_i}\sum_{k=1}^{s} s_k K_o\left(\vec{d_{\text{k,red}}},\vec{d_{\text{i,red}}}\right) \quad . \tag{1.88}$$

1.7 Incorporating Inequalities with Function Values

Another example of application for knowledge about the margin of an example is the incorporation of the additional constraints in the form of inequalities with function values for training points for regression case:

$$g\left(\vec{x_i}\right) \geq a_i \tag{1.89}$$

for $i = 1..n$, where a_i are some parameters, g is defined in (1.24). We propose soft incorporation by changing φ_i values. For ε-SVR, it leads to the modification of the ε_u^i value in OP 7:

$$\varepsilon_{\mathrm{u,new}}^i = \min\left(\varepsilon_{\mathrm{u,old}}^i, y_r^i - a_i\right) \tag{1.90}$$

Proof. From (1.33) we have

$$g\left(\vec{x_i}\right) \geq y_r^i - \varepsilon_u^i - \xi_r^i \tag{1.91}$$

It is a soft incorporation so

$$g\left(\vec{x_i}\right) \geq y_r^i - \varepsilon_u^i \tag{1.92}$$

So we should set ε_u^i such as

$$y_r^i - \varepsilon_u^i \geq a_i \tag{1.93}$$

$$\varepsilon_u^i \leq y_r^i - a_i \tag{1.94}$$

so

$$\varepsilon_{\mathrm{u,new}}^i = \min\left(\varepsilon_{\mathrm{u,old}}^i, y_r^i - a_i\right) \tag{1.95}$$

□

For δ-SVR, we set a value of the δ_d^i parameter in OP 9:

$$\delta_{\mathrm{d,new}}^i = \min\left(\delta_{\mathrm{d,old}}^i, y_r^i - a_i\right) \ . \tag{1.96}$$

Proof. After shifting data the barrier for the function is the shifted point. It will be more restrict barrier, because it is a barrier for the margin boundary. It implies the barrier for the function. We have

$$y_r^i - \delta_d^i \geq a_i \tag{1.97}$$

$$\delta_d^i \leq y_r^i - a_i \ . \tag{1.98}$$

So we modify the shifting parameter δ_d^i as

$$\delta_{\mathrm{d,new}}^i = \min\left(\delta_{\mathrm{d,old}}^i, y_r^i - a_i\right) \ . \tag{1.99}$$

□

Although we do not modify directly φ_i weights, they are modified indirectly for ε-SVR, because changing ε_i leads to changing φ_i. Changing δ_i can be interpreted as changing ε_i.

Similar incorporation scheme exists for

$$g\left(\vec{x_i}\right) \leq a_i \quad . \tag{1.100}$$

For ε-SVR, it leads to the modification of the ε_d^i value in OP 7:

$$\varepsilon_{\mathrm{d,new}}^i = \min\left(\varepsilon_{\mathrm{d,old}}^i, a_i - y_r^i\right) \tag{1.101}$$

Proof. From (1.34) we have

$$g\left(\vec{x_i}\right) \leq \varepsilon_d^i + y_r^i + \xi_r^i \tag{1.102}$$

It is a soft incorporation so

$$g\left(\vec{x_i}\right) \leq \varepsilon_d^i + y_r^i \tag{1.103}$$

So we should set ε_d^i such as

$$\varepsilon_d^i + y_r^i \leq a_i \tag{1.104}$$

$$\varepsilon_d^i \leq a_i - y_r^i \tag{1.105}$$

so

$$\varepsilon_{\mathrm{d,new}}^i = \min\left(\varepsilon_{\mathrm{d,old}}^i, a_i - y_r^i\right) \tag{1.106}$$

□

For δ-SVR, we set a value of the δ_u^i parameter in OP 9:

$$\delta_{\mathrm{u,new}}^i = \min\left(\delta_{\mathrm{u,old}}^i, a_i - y_r^i\right) \tag{1.107}$$

Proof. We have

$$y_r^i + \delta_u^i \leq a_i \tag{1.108}$$

$$\delta_u^i \leq a_i - y_r^i \tag{1.109}$$

So we modify the shifting parameter δ_u^i as

$$\delta_{\mathrm{u,new}}^i = \min\left(\delta_{\mathrm{u,old}}^i, a_i - y_r^i\right) \quad . \tag{1.110}$$

□

We can also increase values of proper C_i parameters to improve fulfillment of the constraints.

Another incorporation type is of the same form as (1.89) but defined for a new point $\vec{x_{n+1}}$ without defined y_r^{n+1}

$$g\left(\vec{x_{n+1}}\right) \geq a_{n+1} \quad , \tag{1.111}$$

where a_{n+1} is a parameter, g is defined in (1.24). We propose soft incorporation. We will use a special version of ε-SVR, where we allow presence of only one constraint either (1.33) or (1.34). So for (1.111) we will have only one constraint that is (1.33). We set

$$y_r^{n+1} = a_{n+1} \tag{1.112}$$

After substituting above to (1.90) we get

$$\varepsilon_{\text{u,new}}^{n+1} = \min\left(\varepsilon_{\text{u,old}}^{n+1}, 0\right) \tag{1.113}$$

One constraint in the formulation means that while transforming to the φ-SVC only one classification point will be created, in this case with $y_c^{n+1} = 1$.

For δ-SVR, we propose also soft incorporation. We will use a special version of δ-SVR, where we allow presence of points which are not duplicated but shifted either up or down. We define such point with $y_c^{n+1} = -1$ and we set

$$y_r^{n+1} = a_{n+1} \tag{1.114}$$

and we shift it down by

$$\delta_{\text{d,new}}^{n+1} = \min\left(\delta_{\text{d,old}}^{n+1}, 0\right) \quad . \tag{1.115}$$

For

$$g\left(\vec{x_{n+1}}\right) \leq a_{n+1} \quad , \tag{1.116}$$

For (1.116) we will have only one constraint that is (1.34). We set

$$y_r^{n+1} = a_{n+1} \tag{1.117}$$

After substituting above to (1.101), we get

$$\varepsilon_{\text{d,new}}^{n+1} = \min\left(\varepsilon_{\text{d,old}}^{n+1}, 0\right) \tag{1.118}$$

One constraint in the formulation means that while transforming to the φ-SVC only one classification point will be created, in this case with $y_c^{n+1} = -1$.

For δ-SVR, we define a point with $y_c^{n+1} = 1$ and we set

$$y_r^{n+1} = a_{n+1} \tag{1.119}$$

and we shift it up by

$$\delta_{\text{u,new}}^{n+1} = \min\left(\delta_{\text{u,old}}^{n+1}, 0\right) \quad . \tag{1.120}$$

Chapter 2

Order Execution Strategies

Big orders cannot be executed on exchanges at once because of the limited number of offers on the opposite side. They must be split into smaller orders and execute in a longer time period. There are various possible measures of the quality of order execution. The most popular are market volume-weighted average price (VWAP), pre-trade price, and post-trade price, all values compared to VWAP for the order. In this report, we investigate the first one. The model of the strategy achieving market VWAP was presented recently in [1, 2]. Bialkowski et al. [1] found that improving quality of the volume prediction leads to better execution performance, however they had found contradicting results in [4].

The goal of the conducted work was to extend the theoretical results for the execution strategy achieving VWAP and to show on which factors the final execution error depends on. Furthermore, we wanted to implement part of the strategy by using a general purpose machine learning method such as SVR. The work was published in [9].

Bialkowski et al. [1] predict a volume function by decomposing volume into two parts and using the average method and autoregressive models for prediction. Brownlees et al. [2] predict a volume participation function by decomposing it to some parts and using a generalized method of moments for predicting parameters of a statistical model. We propose to use a different approach for prediction, namely, use general machine learning methods, which do not assume any particular distribution and statistical properties of the model. We compared SVR with some proposed null hypotheses such as predicting volume participation while assuming constant volume profile, prediction based on average from historical data for the same time slice and prediction from the previous time slice.

The final execution performance depends not only on volume but also on stock prices during order execution. One of ways of improving the strategy is to incorporate information about prices to the model. The presented strategy splits the order to smaller chunks based on volume participation function. The possible way of incorporating information about prices to the model is to adjust volume participation function. We propose modeling the final solution by incorporating prior knowledge about prices by using knowledge about the margin of an example, recently proposed for SVC, [5], for δ-SVR, [6] and for ε-SVR, [8]. It was used for manipulating a decision curve for classification problems, and manipulating a regression function for regression ones.

A test scenario that is investigated in this article is to split execution of the order during a one exchange session. Note that the size of the order has a direct influence on the possibility of achieving VWAP. It is easier to achieve VWAP for bigger orders relative to the daily volume, because the order is also part of the market VWAP. In the extreme situation where the order is the only one executed during the session we achieve VWAP (neglecting transaction costs of executing the order).

The outline is as follows. In the first section, we define VWAP ratio, in the second section, we present an introduction to Volume Participation Strategy. In the third section, we present predicting volume participation, in the fourth section, we show how to incorporate prior knowledge about prices, and finally in the fifth section, we describe conducted experiments.

2.1 VWAP Ratio

In this section, we present the definition of the VWAP ratio, preceded by some definitions and statements regarding VWAP measure, which we will use later. First, we will introduce some notation: T is the time period for executing the order (e.g. one session), n is the number of trades during T, $v(i)$ is a volume of the i-th trade, v is a market volume in T, $p(i)$ is a price of the i-th trade. We have

$$v = \sum_{i=1}^{n} v(i) \ . \tag{2.1}$$

Definition 2.1.1 (Market VWAP)**.** Market VWAP is

$$VWAP = \frac{\sum_{i=1}^{n} p(i)\, v(i)}{v} \ . \tag{2.2}$$

For a volume of the order in T, labeled v_0, we have

$$v_0 = \sum_{i=1}^{n} v_0(i) \ , \tag{2.3}$$

where $v_0(i)$ is part of the order volume belongs to the i-th trade.

Definition 2.1.2 (Order VWAP)**.** Order VWAP is

$$VWAP_0 = \frac{\sum_{i=1}^{n} p(i)\, v_0(i)}{v_0} \ . \tag{2.4}$$

In the presented strategy, we divide T to some time slices. Below, we list some statements regarding time slices.

Proposition 2.1.1. *Assuming that the volume is divided to two parts with known $VWAP$ for these parts ($VWAP_1$ and $VWAP_2$) and known volumes (v_1 and v_2 respectively), overall $VWAP$ is*

$$VWAP = \frac{VWAP_1 v_1 + VWAP_2 v_2}{v_1 + v_2} \ . \tag{2.5}$$

We can generalize this proposition to multiple parts, e.g. multiple time slices, we can divide T to m parts, aggregated volume from all trades in the i-th part is noted as $v\left(T_i\right)$, $VWAP$ for all trades in the i-th part is noted as $VWAP\left(T_i\right)$, aggregated volume of the order in the i-th part is noted as $v_0\left(T_i\right)$. Then a market volume in T is

$$v = \sum_{i=1}^{m} v\left(T_i\right) \ . \tag{2.6}$$

Market VWAP in T is

$$VWAP = \frac{\sum_{i=1}^{m} VWAP\left(T_i\right) v\left(T_i\right)}{v} \ . \tag{2.7}$$

A volume of the order in T is

$$v_0 = \sum_{i=1}^{m} v_0\left(T_i\right) \tag{2.8}$$

and order VWAP in T is

$$VWAP_0 = \frac{\sum_{i=1}^{m} VWAP_0\left(T_i\right) v_0\left(T_i\right)}{v_0} \ . \tag{2.9}$$

In this report, we investigate a problem of developing a strategy that optimizes the ratio of order VWAP to market VWAP for future trades.

Definition 2.1.3 (VWAP ratio)**.** A VWAP ratio is defined as

$$\frac{VWAP_0}{VWAP} = \frac{\sum_{i=1}^{n} p\left(i\right) v_0\left(i\right)}{v_0} \frac{v}{\sum_{i=1}^{n} p\left(i\right) v\left(i\right)} \ . \tag{2.10}$$

We can reformulate (2.10) by substituting

$$v_0 = V_1 v \tag{2.11}$$

and we get

$$\frac{VWAP_0}{VWAP} = \frac{\sum_{i=1}^{n} p\left(i\right) v_0\left(i\right)}{V_1 \sum_{i=1}^{n} p\left(i\right) v\left(i\right)} \ , \tag{2.12}$$

where V_1 is a ratio of order volume to market volume

$$V_1 = \frac{v_0}{v} \ . \tag{2.13}$$

For m time slices we get

$$\frac{VWAP_0}{VWAP} = \frac{\sum_{i=1}^{m} VWAP\left(T_i\right) v_0\left(T_i\right)}{V_1 \sum_{i=1}^{m} VWAP\left(T_i\right) v\left(T_i\right)} \quad . \tag{2.14}$$

For buy orders we would like to minimize this ratio, for sell orders maximize. Particularly, the goal is to achieve the ratio equal or less than 1 for buy orders and equal or greater than 1 for sell orders. Note that a challenge in optimizing this ratio is that future volume and/or future prices have to be predicted. First, we will present a strategy that achieves the ratio equal to 1 by predicting volume participation. Second, we will present an extension of this strategy that allows to incorporate information about prices. Such separation is desirable, because we can compute the error for prediction based on volume, and the error for price prediction.

2.2 Volume Participation Strategy

Here, we describe a model of the strategy that achieves VWAP ratio equal to 1 without assuming any price information. The strategy is to trade with a predicted volume. It means that for every time slice T_i we have

$$v_0\left(T_i\right) = V_1 v\left(T_i\right) = \frac{v_0}{v} v\left(T_i\right) \quad . \tag{2.15}$$

We can see that the strategy satisfies (2.8). We can reformulate it

$$v_0\left(T_i\right) = \frac{v\left(T_i\right)}{v} v_0 = r\left(T_i\right) v_0 \quad , \tag{2.16}$$

where

$$r\left(T_i\right) = \frac{v\left(T_i\right)}{v} \quad . \tag{2.17}$$

The r is called *volume participation*, Fig. 2.1. We can easily check that for this strategy (2.11) is satisfied. After substituting (2.15) to (2.14) we get the ratio equals to 1.

In order to use this strategy in practice we have to predict volume participation $r\left(T_i\right)$ (2.16) for every time slice and try to trade at $VWAP\left(T_i\right)$ inside every time slice. Note that it would be possible to use (2.15) instead of (2.16), but then we would need to predict volume v. Predicting separately volume v and $v\left(T_i\right)$ is more richer prediction than just only ratios $r\left(T_i\right)$. For the same ratios, we could have multiple possible values of v. In other words, when we have only ratios $r\left(T_i\right)$ it is impossible to conclude about a value of v. There exist multiple different volume shapes with the same ratios $r\left(T_i\right)$.

Note that for different values of a free term a of a volume function we can get different values of $r\left(T_p\right)$ for some p, in other words translating a volume function would change $v_0\left(T_p\right)$

$$v_0\left(T_p\right) = \frac{v_0\left(v\left(T_p\right) + a\right)}{\sum_{i=1}^{m} v\left(T_p\right) + a} \quad . \tag{2.18}$$

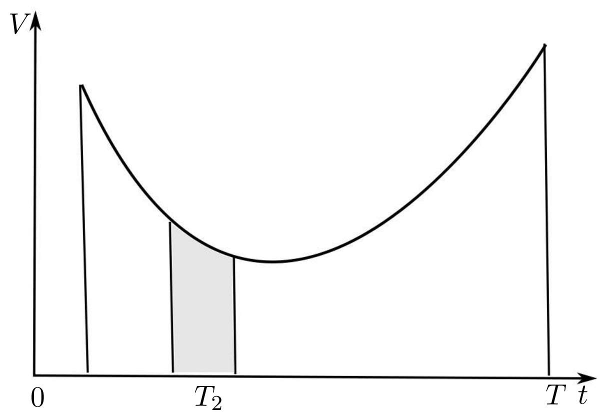

Figure 2.1: The idea of volume participation. Volume participation for T_2 is interpreted as a ratio of gray area to the whole area below volume from 0 to T

The $v_0\left(T_p\right)$ can have different values for different values of the free term a. So it is not enough to predict only volume shape (without a free term).

Let's consider an improvement to the model that our orders are taken into account in global volume. We will redefine v as a volume of other orders. Then we have

$$VWAP = \frac{\sum_{i=1}^{m} VWAP\left(T_i\right)\left(v\left(T_i\right)+v_0\left(T_i\right)\right)}{v+v_0} \quad . \tag{2.19}$$

For m time slices the ratio is

$$\frac{VWAP_0}{VWAP} = \frac{\left(v+v_0\right)\sum_{i=1}^{m} VWAP\left(T_i\right)v_0\left(T_i\right)}{v_0\sum_{i=1}^{m} VWAP\left(T_i\right)\left(v\left(T_i\right)+v_0\left(T_i\right)\right)} \quad . \tag{2.20}$$

Let's analyze the similar strategy of trading as before, that is

$$v_0\left(T_i\right) = \frac{v_0}{v}v\left(T_i\right) \quad . \tag{2.21}$$

We can see that (2.8) is satisfied. Let's derive the ratio

$$\frac{VWAP_0}{VWAP} = \frac{\left(v+v_0\right)\frac{v_0}{v}\sum_{i=1}^{m} VWAP\left(T_i\right)v\left(T_i\right)}{v_0\left(1+\frac{v_0}{v}\right)\sum_{i=1}^{m} VWAP\left(T_i\right)v\left(T_i\right)} \tag{2.22}$$

$$\frac{VWAP_0}{VWAP} = \frac{v_0+\frac{v_0^2}{v}}{v_0+\frac{v_0^2}{v}} = 1 \quad . \tag{2.23}$$

We can see that again a VWAP ratio is equal to 1.

2.2.1 Errors for Volume Participation Strategy

There are two possible sources of execution errors in this strategy. The first error ε_1 is related to trading with $VWAP\left(T_i\right)$, the second error ε_2 is related to predicting volume participation in T_i, after substituting (2.16) to (2.9) and considering the errors

$$VWAP_0 = \sum_{i=1}^{m} \left(VWAP\left(T_i\right) + \varepsilon_1\left(T_i\right)\right)\left(r\left(T_i\right) + \varepsilon_2\left(T_i\right)\right) \ . \tag{2.24}$$

While comparing $VWAP$ to $VWAP_0$ we get the following error (derivation in A.1)

Theorem 2.2.1.

$$\varepsilon = \frac{VWAP_0}{VWAP} - 1 = \frac{\sum_{i=1}^{m} \varepsilon_1\left(T_i\right) r\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} + \frac{\sum_{i=1}^{m} \varepsilon_2\left(T_i\right) VWAP\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} \tag{2.25}$$

$$+ \frac{\sum_{i=1}^{m} \varepsilon_1\left(T_i\right) \varepsilon_2\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} \ . \tag{2.26}$$

In this report, we are interested mainly in optimizing ε_2. So we either generate prior values of E_1 where $\varepsilon_1\left(T_i\right) = E_1\left(T_i\right) VWAP\left(T_i\right)$, or substitute $\varepsilon_1\left(T_i\right) = 0$. Lowering ε_2 leads to a lower variance of ε.

Comparison to time-weighted average price (TWAP) strategy. The TWAP strategy trades the same quantity in every time slice. The TWAP can be interpreted as the volume participation strategy with predicted volume as a constant function. We expect worse performance of prediction of volume participation for TWAP, therefore larger value of ε_1 compared to the VWAP strategy, so we expect larger variance of ε for the TWAP method.

2.3 Predicting Volume Participation

In order to use Volume Participation Strategy we need to predict volume participation $r\left(T_i\right)$ for all time slices. In this report, we investigate four methods of prediction, the first one arbitrarily assumes that a volume is a constant function, so a volume participation function is also a constant one (it is used in the TWAP strategy), the second one predicts $r\left(T_i\right)$ as an average value from previous days, it is kind of a local strategy. The third one predicts $r\left(T_i\right)$ as $r\left(T_{i-1}\right)$ ($r\left(T_{i-1}\right)$ is predicted as in the second solver) and the last one predicts volume participation $r\left(T_i\right)$ from historical data by assuming that $r\left(\cdot\right)$ is a continuous function. There is only one feature that is the id of the time slice, so the feature space is a discrete one. For the last prediction, we use SVR methods. Volume participation prediction has two additional constraints that should be satisfied:

$$\sum_{i=1}^{m} r\left(T_i\right) = 1 \ , \tag{2.27}$$

$$r\left(T_i\right) > 0 \ . \tag{2.28}$$

For the TWAP predictor, they are satisfied out of hand. For the remaining predictors we need special consideration. For the second predictor, we propose the following procedure:

we equally decrease values of all $r(T_i)$ in order to satisfy (2.27), and when some values are below zero, we adjust them to zero. We repeat these two steps until both constraints are satisfied. For the last predictor, we propose the direct incorporation of (2.27) by using φ-SVC and modified kernels, 1.6. Instead of incorporating directly (2.28) to the optimization problem, we propose soft incorporation proposed for SVC, 1.7.

2.4 Incorporating Prior Knowledge About Prices

Volume Participation Strategy achieves the ratio equal to 1 in the presented model. It is possible to achieve better execution performance by taking into account price prediction. The general idea of an improvement is to increase order volume when the predicted price is relatively low during the session for buy orders (relatively high for sell orders).

There are two problems concerning manipulating a participation function based on price prediction. First is in achieving enough price prediction performance for improving the error ε. Second, that increased order volume for some time slices could change noticeably the prices during the next sessions (it is called *market impact*) and additionally decrease price prediction performance.

Because price prediction is a challenging task, we propose to incorporate simple price prediction rules, such as *in the second part of the session prices will be higher than in the first one (or vice versa).* For this rule we might want to increase participation in the first half of the session, and decrease in the second one (for buy orders). The simple way of incorporating such knowledge is to increase participation by some value e.g. $p = 0.1$ for the first part of the session and decrease by the same value in the second part of the session (assuming the even number of time slices). The problem with this solution is that participation rate is not smooth in the half of the session. The second issue is that participation changes by the same value in the first part and the second. We cannot improve participation changes by using price information, because we have just only simple prediction rules. So we propose to set participation changes based on volume participation prediction performance. We want to increase value and chance of changing p for time slices with worse volume participation prediction performance, and decrease value of p for the rest. For this purpose, we use SVM with knowledge about the margin of an example introduced for SVC in [5, 6], for ε-SVR in [8] and for δ-SVR in [7]. The technique was used for manipulating classification boundaries, [5], and regression functions, [7]. It has a desired property of adjusting the output function depending on the prediction performance.

2.4.1 Defining Knowledge About Prices

We divide the period T to 2 periods, first half of the session and the second. We propose setting $\varphi_i = r$ for all training examples, where r is a configurable parameter. When we expect that prices will be higher in the second part of the session, for every example from the first part of the session we set -1 class, and for the second part we set 1 class (in reverse for opposite prediction).

2.5 Experiments

We divide experiments into three parts: in the first part we compare prediction performance of SVM with null hypotheses. In the second experiment, we compare execution error for SVM and null hypotheses. We compare prediction performance of SVM with the following null hypotheses: prediction based on a constant function, prediction based on average participation from historical data for the same time slice and prediction from the previous time slice. In the third experiment, we compare ε for δ-SVR and δ-SVR with incorporated knowledge about the margin of an example.

For solving ε-SVR and SVC for particular values of parameters we use LibSVM, [3], ported to Java. Data that are used for experiments are tick data for securities from National Association of Securities Dealers Automated Quotations (NASDAQ)-100 index for about a half year period (from 01.01.2011 to 20.05.2011), which were compressed to a desired size of time slices. Data include trades from opening and closing crosses. For all data sets, every feature is scaled linearly to $[0, 1]$. The results are averaged for all tested instruments. For variable parameters like the C, σ for the RBF kernel, δ for δ-SVR, and ε for ε-SVR, we use a double grid search method for finding the best values. We use modified double cross-validation with shifting data. Inner cross-validation is used for finding the best values of the variable parameters. Instead of standard outer cross-validation, we shift data. Hence, the validation set is always after the training set. We use a fixed size for the training set, that is 2 weeks, and for the validation set 1 week.

2.5.1 Prediction Performance and Error Comparison

We compare δ-SVR and ε-SVR with null hypotheses. Results are presented in Table 2.1. For fair comparison purposes we choose $\varepsilon_1 = 0$. We performed tests for half hour slices.

We achieve better generalization performance for ε-SVR and δ-SVR for almost all null hypotheses with better results for δ-SVR. The ε-SVR had problems with achieving significant improvements for a linear kernel. The average null hypothesis is the most competitive comparing to SVR, we achieve slightly better generalization performance for SVR, but without significant difference based on t-test for ε-SVR, with significant difference for δ-SVR. Comparing additional measure of variance of execution error, we achieve slightly better results for ε-SVR than for the first and the third hypotheses, and similar results to the second hypothesis. For δ-SVR, we achieved much larger variance of ε then for all hypotheses.

2.5.2 Execution Performance with Knowledge About Prices

We compare ε for δ-SVR with incorporated prior knowledge about prices, and without. The scope of this report omits the topic of price prediction. Therefore, we propose the following procedure for generating prior knowledge about prices, we check in advance on historical data whether market VWAP will be higher in the first part of the session, or in the second one. According to this prediction we set φ_i weights, r value is chosen arbitrarily to 0.5. Results are presented in Table 2.2.

Table 2.1: Performance of δ-SVR for order execution. Column descriptions: *id* – an id of a test, *a name* – a name of the test, δ-SVR compared with hypotheses 1 or 2 or 3, *ts* – a size of time slice (in minutes), *simT* – the number of shifts, results are averaged, *ker* – a kernel (*pol* – a polynomial kernel), *kerP* – a kernel parameter (for a polynomial kernel it is a dimension, for the RBF kernel it is σ), *trs* – a training set size for every stock, *all* – the number of all data for every stock, *dm* – a dimension of the problem, *tr12M* – a percent average difference in mean error for training data, if greater than 0 than SVM is better, *te12M* – the same as tr12M, but for testing data, *teT* – t value for the t-test for comparing testing error, *e12M* – comparison of a variance of ε. The value 'var' means that we search for the best value

id	name	ts	simT	ker	kerP	trs	all	dm	tr12M	te12M	teT	e12M
1	δ-SVRvsH1	30m	5	lin	—	130	1075	1	12.7%	11.7%	15.2	−92.6%
2	ε-SVRvsH1	30m	5	lin	—	130	1075	1	1.11%	0.7%	0.8	0.23%
5	δ-SVRvsH1	30m	5	rbf	0.1	130	1075	1	51.2%	46.9%	62.6	-72.1%
6	ε-SVRvsH1	30m	5	rbf	0.1	130	1075	1	49.5%	45.6%	59.9	0.28%
11	δ-SVRvsH2	30m	5	rbf	0.1	130	1075	1	2.75%	3.4%	3.0	−72%
12	ε-SVRvsH2	30m	5	rbf	0.1	130	1075	1	−0.5%	1.17%	1.0	−0.02%
13	δ-SVRvsH3	30m	5	lin	—	130	1075	1	10.83%	9.1%	9.58	−92.5%
14	ε-SVRvsH3	30m	5	lin	—	130	1075	1	−1.05%	−2.13%	−2.1	0.96%
17	δ-SVRvsH3	30m	5	rbf	0.1	130	1075	1	50.1%	45.3%	48.6	-71.9%
18	ε-SVRvsH3	30m	5	rbf	0.1	130	1075	1	48.4%	44.06%	46.7	1.02%

Table 2.2: Performance of δ-SVR with prior knowledge about prices for order execution. Column descriptions: *id* – an id of a test, *ts* – a size of time slice (in hours), *simT* – the number of shifts, results are averaged, *ker* – a kernel (*pol* – a polynomial kernel), *kerP* – a kernel parameter (for a polynomial kernel it is a dimension, for the RBF kernel it is σ), *trs* – a training set size for every stock, *all* – the number of all data for every stock, *dm* – a dimension of the problem, *r* – φ_i value, *tr12M* – a percent average difference in mean error for training data, if greater than 0 than SVM is better, *te12M* – the same as tr12M, but for testing data, *teT* – t value for the t-test for comparing testing error, *e12M* – comparison of ε, *eT* – t-value for comparing ε. The value 'var' means that we search for the best value

id	ts	simT	ker	kerP	trs	all	dm	r	tr12M	te12M	teT	e12M	eT
22	30m	5	rbf	0.1	130	1075	1	1	−5%	−6%	−1.7	19%	2.4

The results show that volume participation prediction performance could be worse after adjusting the function, but we can see significant improvement in execution error for the modified solution. The δ-SVR with prior knowledge about prices achieves better execution performance than without prior knowledge.

2.6 Summary

In this report, we analyzed application of SVR for executing orders on stock markets. We compared ε-SVR and δ-SVR with simple predictors such as the average execution price from previous days for predicting volume participation function. We can improve costs of order execution by using prediction of stock prices with SVM.

In future research, we plan to perform tests on a broader list of stocks and exchanges.

Appendix A

A.1 Proof of Thm. 2.2.1

Proof. The proof is

$$VWAP_0 = \frac{\sum_{i=1}^{m}\left(VWAP\left(T_i\right)+\varepsilon_1\left(T_i\right)\right)\left(v_0\left(r\left(T_i\right)+\varepsilon_2\left(T_i\right)\right)\right)}{v_0} \tag{A.1}$$

$$VWAP_0 = \sum_{i=1}^{m}\left(VWAP\left(T_i\right)+\varepsilon_1\left(T_i\right)\right)\left(r\left(T_i\right)+\varepsilon_2\left(T_i\right)\right) \tag{A.2}$$

$$\frac{VWAP_0}{VWAP} - 1 = \frac{v\sum_{i=1}^{m}\left(VWAP\left(T_i\right)+\varepsilon_1\left(T_i\right)\right)\left(r\left(T_i\right)+\varepsilon_2\left(T_i\right)\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) v\left(T_i\right)} - 1 = \tag{A.3}$$

$$= \frac{\sum_{i=1}^{m}\left(VWAP\left(T_i\right)+\varepsilon_1\left(T_i\right)\right)\left(r\left(T_i\right)+\varepsilon_2\left(T_i\right)\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} - 1 = \tag{A.4}$$

$$= \frac{\sum_{i=1}^{m}\varepsilon_1\left(T_i\right) r\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} + \frac{\sum_{i=1}^{m}\varepsilon_2\left(T_i\right) VWAP\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} \tag{A.5}$$

$$+ \frac{\sum_{i=1}^{m}\varepsilon_1\left(T_i\right)\varepsilon_2\left(T_i\right)}{\sum_{i=1}^{m} VWAP\left(T_i\right) r\left(T_i\right)} \quad . \tag{A.6}$$

□

References

[1] Jedrzej Bialkowski, Serge Darolles, and Gaelle Le Fol. Improving vwap strategies: A dynamic volume approach. *Journal of Banking & Finance*, 32(9):1709–1722, September 2008. 17

[2] Christian T. Brownlees, Fabrizio Cipollini, and Giampiero M. Gallo. Intra-daily volume modeling and prediction for algorithmic trading. Econometrics working papers archive, Universita' degli Studi di Firenze, Dipartimento di Statistica "G. Parenti", February 2009. 17

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 24

[4] D. Hobson. Vwap and volume profiles. *Journal of Trading*, 1(2):38–42, 2006. 17

[5] Marcin Orchel. Incorporating detractors into svm classification. In Krzysztof Cyran, Stanislaw Kozielski, James Peters, Urszula Stańczyk, and Alicja Wakulicz-Deja, editors, *Man-Machine Interactions*, volume 59 of *Advances in Intelligent and Soft Computing*, pages 361–369. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-00562-6. doi: 10.1007/978-3-642-00563-3_38. URL `http://dx.doi.org/10.1007/978-3-642-00563-3_38`. 9, 17, 23

[6] Marcin Orchel. Incorporating a priori knowledge from detractor points into support vector classification. In Andrej Dobnikar, Uroš Lotric, and Branko Šter, editors, *Adaptive and Natural Computing Algorithms*, volume 6594 of *Lecture Notes in Computer Science*, pages 332–341. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-20266-7. doi: 10.1007/978-3-642-20267-4_35. URL `http://dx.doi.org/10.1007/978-3-642-20267-4_35`. 9, 17, 23

[7] Marcin Orchel. Regression based on support vector classification. In Andrej Dobnikar, Uroš Lotric, and Branko Šter, editors, *Adaptive and Natural Computing Algorithms*, volume 6594 of *Lecture Notes in Computer Science*, pages 353–362. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-20266-7. doi: 10.1007/978-3-642-20267-4_37. URL `http://dx.doi.org/10.1007/978-3-642-20267-4_37`. 8, 23

[8] Marcin Orchel. Support vector regression as a classification problem with a priori knowledge in the form of detractors. In Tadeusz Czachorski, Stanislaw Kozielski, and Urszula Stańczyk, editors, *Man-Machine Interactions 2*, volume 103 of *Advances in Intelligent and Soft Computing*, pages 353–362. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-23168-1. doi: 10.1007/978-3-642-23169-8_38. URL `http://dx.doi.org/10.1007/978-3-642-23169-8_38`. 9, 17, 23

[9] Marcin Orchel. Support vector regression with a priori knowledge used in order execution strategies based on vwap. In Jie Tang, Irwin King, Ling Chen, and Jianyong Wang, editors, *Advanced Data Mining and Applications*, volume 7121 of *Lecture Notes in Computer Science*, pages 318–331. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-25855-8. doi: 10.1007/978-3-642-25856-5_24. URL `http://dx.doi.org/10.1007/978-3-642-25856-5_24`. 17

[10] Ingo Steinwart, Don R. Hush, and Clint Scovel. Training svms without offset. *Journal of Machine Learning Research*, 12:141–202, 2011. 3

[11] Francis Eng Hock Tay and Lijuan Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48(1-4):847–861, 2002. 6

[12] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998. ISBN 0471030031. 6

[13] Xiaoyun Wu and Rohini Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326–333, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. 9

Notation and Symbols

Miscellaneous

$|A|$ the cardinality of a finite set A, i.e., the number of elements in the set A.

$\cdot$ Dot product of two vectors, sometimes it is written with additional parentheses, for example for two vectors: $\vec{u}$ and $\vec{v}$, the dot product is $\vec{u} \cdot \vec{v}$ or $(\vec{u} \cdot \vec{v})$.

$\vec{v} \geq \vec{w}$ For two n dimensional vectors $\vec{v}$ and $\vec{w}$, it means that for all $i = 1...n$ $v_i \geq w_i$.

$\vec{v} \gg \vec{w}$ For two n dimensional vectors $\vec{v}$ and $\vec{w}$, it means that for all $i = 1...n$ $v_i > w_i$.

$\rho\,(A)$ the rank of a matrix A.

$\vec{w}^{i}_{\mathbf{r}}$ When a vector has an index in the subscript, the coefficient index is placed in the superscript, the example means the i-th coefficient of the $\vec{w}_{\mathrm{r}}$.

Optimization theory

* an asterisk as a superscript in optimization theory denotes a solution of the optimization problem.

Abbreviations

δ-SVR	δ support vector regression.
ν-SVC	ν support vector classification.
ε-SVR	ε-insensitive support vector regression.
φ-SVC	φ support vector classification.
C-SVC	C support vector classification.
NASDAQ	National Association of Securities Dealers Automated Quotations.
RBF	radial basis function.
SMO	sequential minimal optimization.
SVC	support vector classification.
SVM	support vector machines.
SVR	support vector regression.
TWAP	time-weighted average price.
VWAP	volume-weighted average price.