"Science Stays True Here"
Journal of Mathematics and Statistical Science, 319-333 | Science Signpost Publishing

Kernel Optimization Techniques for Price Prediction

Hayato Kijima[1], Hideyuki Takada[2]

1. Graduate School of Information Science, Toho University, Miyama 2-2-1, Funabashi, 274-8510, Chiba, Japan

2. Department of Information Science, Toho University, Miyama 2-2-1, Funabashi, 274-8510, Chiba, Japan

Abstract

In this paper, we employ Support Vector Machine to predict future directions of the Nikkei 225 futures by learning from the dynamics of Limit Order Book. In order to improve its accuracy, as our previous paper Kijima and Takada (2017) reported, we apply the method of conformal transform of the kernel function pioneered by Amari and Wu (1999). For comparison we also apply Fisher Criteria based data-dependent kernel optimization method proposed by Xiong, Samy and Ahmad (2005) to evaluate their performance. In this sense the paper is a companion to Kijima and Takada (2017) and we conclude, by comparing empirical results, that the conformal transform of Amari and Wu with ex-ante calibrated model parameters improved the precision more than 3.5% in average compared to the standard Gaussian kernel, while the method of Xiong, Samy and Ahmad improved only 1.5% in average.

Keywords: Limit Order Book; Support Vector Machine; Conformal transformation; Empirical feature space

Introduction

Recently, machine learning technics are often applied to the analysis of Limit Order Book (LOB - described below) to learn future direction of asset prices both for short term (seconds) and long term (daily). For example, Logistic regression analysis in Ban Zheng [1] and Support Vector Machine in Kercheval and Zhang [2] and Deng, Sakurai and Shioda [3] aspired to predict future price directions of some assets. The other machine learning techniques such as Artificial Neural Network, Random forest and Naive-Bayes classifier and Support Vector Regression are applied to predict daily return of stock price without using the information of the LOB as in Patel, Shah, Thakkar and Kotecha [4]. Although it is growing the significance of interests for high frequency data analysis, it seems that the number of published research paper is not so increased because of the confidentiality in industry.

In this paper, we employ Support Vector Machine (SVM - described below) combined with conformally transformed Gaussian kernel to predict the mid-price dynamics in the LOB. For this purpose, as Fletcher and Shawe-Tayjor [5] studied, we first translate the financial language of the LOB into the language of SVM for mathematical formulation. More precisely, the shape of the LOB at each time is

Corresponding author: Hideyuki Takada, Department of Information Science, Toho University, Miyama 2-2-1, Funabashi, 274-8510, Chiba, Japan. E-mail: hideyuki.takada@is.sci.toho-u.ac.jp.

translated to a training data and its outcome (direction of the mid-price movement), which is observable in the future is treated as a corresponding label. We focus on the first moving time of the mid-price rather than the price movement of fixed time interval. Therefore, we formulate our model as a two class (up or down) classifier rather than three class (up, stay and down). We expect that if we can train the SVM with many training data effectively, then the SVM can predict future direction of the mid-price successfully.

In order to improve accuracy of the SVM, we applied kernel modification method pioneered by Amari and Wu [6] and Wu and Amari [7]. The detailed description of the application could be already found in our previous paper Kijima and Takada [8], however, we rephrase anew our methods for readers' convenience. Amari and Wu considered the feature space as a Riemannian manifold, which can be realized as a curved surface embedded in high dimensional Euclidian space, and express the distance of two distinct points in feature space via Riemannian metric. Since the Riemannian metric and the kernel function are related to each other as shown precisely later, we can consider modification of the kernel functions within this framework. The issue is how to modify the kernel function preserving computational tractability. In order to focus on the effect of kernel modification, we don't discuss about sequential updating for control parameters of SVM.

The other stream of conformal transform methods is first proposed by Xiong, Samy and Ahmad [9] and developed to multi-class classifier by Lin, Jiang, Zhao, Pang and Ma [10]. These two studies adhere fundamentally to Amari and Wu [6] and Wu and Amari [7] but attempt to pre-optimize the control variables of conformal transform so as to fit (in the sense that the discriminant function can separate effectively) to given the training data set. The optimization, based on the Fisher Criteria for measuring the class linear separability, is achieved and then we can know the best control variables of conformal transform before the learning process of SVM. One important aspect would be summarized as follows; Amari and Wu method find the optimal control variable via ex-post tuning, while Xiong, Samy and Ahmad method find the optimal control variable ex-ante. Our previous work, Kijima and Takada [8], already pointed out the effectiveness of Amari and Wu method, the above aspect motivate us to compare which method would outperform.

As some existing researches such as Wu and Amari [7], Williams, Li, Feng and Wu [11] report, in the realm of natural science, SVM with conformally transformed Gaussian kernel function exhibit more than 3.5% higher performance than standard Gaussian kernel. Likewise, Xiong, Samy and Ahmad [9] and Lin, Jiang, Zhao, Pang and Ma [10] exhibit substantial improvement of the performance not only in the realm of natural science, but also in the image recognition. Main contribution of our study is to show the same aspect in the area of financial market possibly containing many kinds of noise in the sense that even perfectly same shape of the LOB may produce opposite outcome.

Data

Our empirical analyses are based on the high frequency historical data records of the LOB of Nikkei 225 futures listed in Osaka exchange in Japan. This LOB data is comprised of agreed prices and event records such as volume of limit sell/buy orders for each price level with approximately 20 milliseconds time scale. Instead of these original data sets, we use adequately processed data such as second-scale

historical data obtained by extracting every second from the original data. Example of the LOB data are shown in Table 1 illustrating the best-bid price at time $t =$ 09:02:42.379 is 1012δ = 10120 yen and the best-ask price at the same time is $1013\,\delta$ = 10130 yen. From these records, we can read that the market sell order of size 233 was executed at $t =$ 09:02:47.913 and bid-ask spread of 20 yen continued to appear during the next 644 milliseconds. After that, no limit buy order at $1012\,\delta$ arrived and limit sell order of size 422 at the price $1012\,\delta$ newly reached hence the mid-price changed and bid-ask spread shrunk to normal size, i.e., 10 yen.

Table 1: Example of the LOB data flow of Nikkei 225 futures as of Dec. 25, 2012. (δ = 10 yen)

time	midprice	1008δ	1009δ	1010δ	1011δ	1012δ	1013δ	1014δ	1015δ	1016δ
09:02:42.379	10125	···	···	798	487	237	806	1041	843	···
09:02:43.923	10125	···	···	798	488	233	807	1041	843	···
09:02:47.913	10120	···	···	798	488	0	807	1041	843	···
09:02:48.010	10120	···	830	798	355	0	807	1041	843	···
09:02:48:557	10115	···	830	798	355	422	852	1041	···	···
09:02:48:877	10115	···	830	798	355	428	853	1051	···	···

Trading time of Nikkei 225 futures is divided into following four sessions; pre-opening session, regular session, pre-closing session and night session. During the pre-opening session starting from 8 AM and ending at 9 AM, and during the pre-closing session from 3:00 PM to 3:15 PM, limit orders are received but matching cannot be executed. During the night session from 4:30 PM to 3 AM, Japanese investors do not trade much and then very few transactions are made. Therefore, for our empirical studies, we use the data of regular session starting every weekday from 9 AM and ending 3:10 PM. Our empirical studies are concentrated to the period between April 2, 2012 and June 30, 2012.

State of the Limit Order Book

In order to apply SVM to LOB, we follow the standard formulation of LOB proposed in the previous study by Kijima and Takada [8] as follows. A single snapshot of the LOB can be handled as a high dimensional vector recorded with time-stamp. For more precise description of the LOB, we introduce coordinate system consist of time-axis, price-axis and volume-axis as shown in Figure 1. As trade progress, mid-price will change and then we need to prepare huge space to cover full range of the price dynamics in each day. In order to confine this region to more saving but essentially space, we slide the price-axis and take the mid-price as an origin of the price-axis for each time in this coordinate system. Of course, alternatively we can take best-ask price or a best-bid as an origin of the price-axis, but it is not essential. The reason why we pay attention to the distance between the limit orders between mid-price rather than exact price level of the limit order is, many agents often attach great importance to how deep their orders are. Then each coordinate has three components such as time, the distance between the limit order and mid-price, and the volume size of the limit order. It is assumed that limit orders and market orders can be placed on a fixed price grid $\{1,2,\cdots,N\}$ representing multiples of a price tick denoted by δ, so the state

of the LOB can be seen as a discrete function on a discrete line calibrated with unit length of δ.

The upper boundary N is chosen large enough so that it is highly unlikely that orders at prices higher than N will be placed within the time frame of our analysis[1]. State of the LOB at time t is described by the discrete time N-dimensional stochastic process, where each element denotes the time t order size waiting for the future market order to be matched. Some of them, which are located near the mid-price, would be observable but the others, far from the mid-price, would be hidden. For example, in case of Nikkei 225 futures market, investors cannot necessarily observe all the limit orders although more exists. Because of the narrow window of records, only the information close to the best price (generally 10 prices for both sell and buy orders) are available and limit orders placed far from the best prices are not shown for investors.

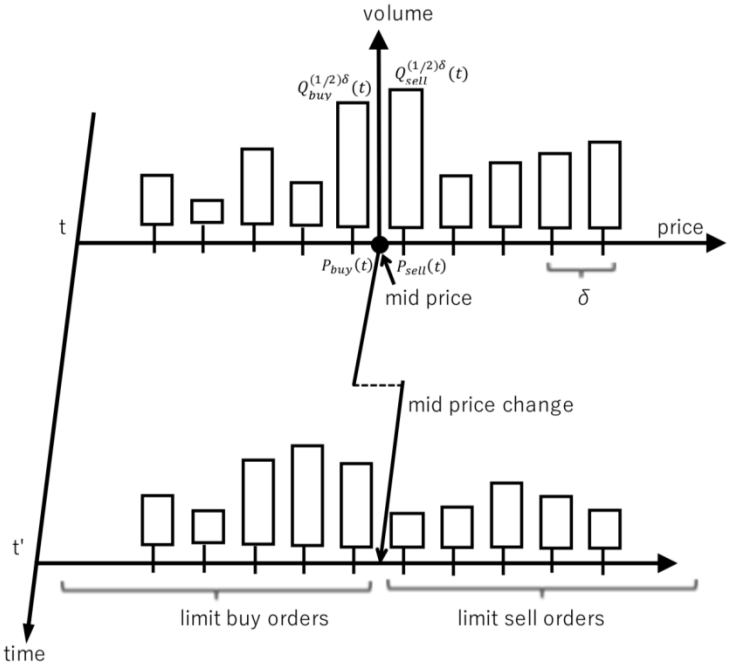

Figure 1: Time evolution of the Limit Order Book

Let the best-ask price at time t is denoted by $P_{sell}(t) > 0$ and similarly the best-bid price is denoted by $P_{buy}(t) > 0$. We define the number of outstanding sell orders at a distance $k + 1/2$, $k = 0,1,2,\cdots$ (equivalently $(k + 1/2) \cdot \delta$ in price[2]) from the mid-price $(P_{sell}(t) + P_{buy}(t))/2$ as $Q_{sell}^{(k+1/2)\delta}(t) \in Z_+$. Thus, the quantity $Q_{sell}^{(1/2)\delta}(t)$ indicates the number of orders of best-ask at time t. Similarly, the number of outstanding buy orders at a distance $k + 1/2$, $k = 0,1,2,\cdots$ from the best-ask price is defined as $Q_{buy}^{(k+1/2)\delta}(t) \in Z_+$. As a general reference, the detailed quantitative descriptions for

[1] As described in Cont, Stoikov and Talreja [12] since the model is intended to be used on the time scale of days, this finite boundary assumption is reasonable.

[2] Here, $1/2$ corresponds to the distance between the mid-price and the best-ask price.

LOB could be found in the manuscript by Martin D. Gould, Mason A. Porter, Stacy Williams, Mark Macdonald, Daniel J. Fenn et al [13].

Application of the Support Vector Machine

In this paper, we attempt to classify short-term (order of second) future price direction with SVM and focus on the improvement of its accuracy. We expect that the shape of the LOB is predictably effective for the direction of the next mid-price movements. As discussed above, the trajectory of the LOB can be seen as high dimensional discrete time stochastic process that records the sequence of volume size of limit orders and the distances from the mid-price, while absolute price levels are not contained. For simple explanation of our methodology, let us first consider the best-price only, and market order will arrive like a random walker as assumed in Gerry Tsoukalas, Jiang Wang, Kay Giesecke [14]. When the volume of the best-ask price is relatively larger than that of the best-bid price, it is reasonable to consider that the market has a high likelihood of downtrend because the best-bid orders may have eaten up by certain amount of the market sell orders in the next instant. This simple intuition would be applicable to the general circumstances considering all available limit orders. In fact, Fletcher and Shawe-Tayjor [5] considered not only the snapshot of the LOB, but also the time-derivatives (these are finite difference because of the discrete time setting) of the volume size at each price level, which contains past information. In our study, we focus on the effectiveness of modification of the kernel function thus the simple structure of training data is preferred.

We consider discretized time grids $T_i, 0 \leq i \leq M$ to express full available historical data records of the LOB and the mid-price. Let the available shape of the LOB, i.e., the order book volume at time T_i is identified with $2(n+1)$-dimensional vector

$$\boldsymbol{x_i} = \left(Q_{buy}^{(n+1/2)\delta}(T_i), \dots, Q_{buy}^{(1/2)\delta}(T_i), Q_{sell}^{(1/2)\delta}(T_i), \dots, Q_{sell}^{(n+1/2)\delta}(T_i)\right).$$

In case of Nikkei 225 futures, constantly available data is restricted to $0 \leq n \leq 8$, which may sound odd. In many cases, we can see limit sell/buy orders placed in the price corresponding to $n = 9$, however, sometimes at the moment of mid-price change, the bid-ask spread that is strictly larger than δ appears in a very short time and then the limit order corresponding to $n = 9$ is pushed out of the records. This is the reluctant reason why we have no other alternative but to handle the data with $0 \leq n \leq 8$.

Let $\pi(T_i) \coloneqq (P_{sell}(T_i) + P_{buy}(T_i))/2$ denote the mid-price at time T_i and define

$$\mathfrak{T}(T_k) = \min\{j | \pi(T_{j+1}) \neq \pi(T_j), j \geq k\}$$

as an index of discretized time grid at which the mid-price moved for the first time after T_k. Training data for SVM is a set of sequences of prior trials $\{\boldsymbol{x_i} \in R^{2(n+1)}\}_{0 \leq i \leq m}$ with $m \leq M$ that have already been classified into two classes. In our case, classification means that each $\boldsymbol{x_i}$ has been assigned a label $y_i \in \{+1, -1\}$ depending on the direction of the first mid-price movement as follows.

$$y_i = \begin{cases} +1 & if\ \pi(T_{\mathfrak{T}(T_i)}) > \pi(T_{\mathfrak{T}(T_i)-1}) \\ -1 & if\ \pi(T_{\mathfrak{T}(T_i)}) < \pi(T_{\mathfrak{T}(T_i)-1}) \end{cases}$$

That is, if the first mid-price movement after T_i was upward direction, then we set $y_i = +1$ and in an

opposite case we set $y_i = -1$.

Nonlinear SVM maps the input data $\boldsymbol{x} \in I = R^{2(n+1)}$ into a higher dimensional feature space $F = R^N$, $2(n+1) \leq N$ by a nonlinear mapping Φ. By choosing an adequate mapping Φ, the data points become mostly linearly separable in the feature space. Trained SVM finds maximum margin hyper plane in a feature space F as a final decision boundary defined by

$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i K(\boldsymbol{x}, \boldsymbol{x_i}) + b$$

where K is called kernel function and summation runs over all the support vectors. Here, parameters α_i are derived by solving quadratic programing problem

$$\max_{\alpha} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \, K(\boldsymbol{x_i}, \boldsymbol{x_j})$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

with prespecified parameter C, which controls the trade-off between margin and misclassification error. It is well known that $\Phi(\boldsymbol{x}) \in F$ is not necessarily known to derive separating boundary and we only need to know the inner product of vectors $\Phi(\boldsymbol{x})$ in the feature space by virtue of $K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}')$. Thus, the trained SVM will return predicted direction $y_{m+1} = h(\boldsymbol{x}_{m+1}) \coloneqq sign(f(\boldsymbol{x}_{m+1})) \in \{+1, -1\}$ for new trial data $\boldsymbol{x}_{m+1}$.

Geometric reformulation of the Kernel

In order to improve the performance of SVM classifiers, Amari and Wu [6] and Wu and Amari [7] proposed conformal transformation of kernel functions based on the understanding that a good kernel should enlarge the separation between the two classes. Furthermore, Williams, Li, Feng and Wu [11] studied more robust method in the sense that the additional free parameter is only one and then computational algorithm is kept simple.

From geometrical point of view, the mapped data $\boldsymbol{z} = \Phi(\boldsymbol{x})$ generally lie on a $2(n+1)$ dimensional surface S in F. Hear we omitted the subscript i in $\boldsymbol{x}_i$ indicating the time recorded for simplicity. If we assume that Φ has all continuous derivatives, S can be seen as an embedded submanifold possessing Riemannian metric. The Riemannian metric enable one to measure the distance of two distinct points on S via line integral along the shortest curve (geodesic) connecting these two points. Here, the line element, denoted by ds, can be expressed as

$$(ds)^2 = \sum_{1 \leq i,j \leq 2(n+1)} g_{ij} dx^i dx^j$$

where g_{ij} is called Riemannian metric and superscript i in x^i denote the i-th element of vector $\boldsymbol{x}$. Applying Φ, $2(n+1)$-dimensional vector $\boldsymbol{dx} = (dx^1, \cdots, dx^{2(n+1)})$ is mapped to

$$\boldsymbol{dz} = \Phi(\boldsymbol{x} + \boldsymbol{dx}) - \Phi(\boldsymbol{x}).$$

Then the line element ds is expressed in feature space as

$$(ds)^2 = \|ds\|^2 = \|\Phi(\boldsymbol{x} + \boldsymbol{dx}) - \Phi(\boldsymbol{x})\|^2.$$

From Taylor's theorem for $2(n+1)$-variate function $R^{2(n+1)} = I \ni \boldsymbol{x} \mapsto \Phi(\boldsymbol{x}) = (\Phi^1(\boldsymbol{x}), \cdots, \Phi^N(\boldsymbol{x})) \in F$, the line element ds is re-expressed with the kernel function K as follows

$$(ds)^2 = \sum_{k=1}^{N} \left\{ \sum_{j=1}^{2(n+1)} \frac{\partial \Phi^k(\boldsymbol{x})}{\partial x^j} \right\}^2$$

$$= \sum_{k=1}^{N} \sum_{1 \le i,j \le 2(n+1)} \frac{\partial \Phi^k(\boldsymbol{x})}{\partial x^i} \frac{\partial \Phi^k(\boldsymbol{x})}{\partial x^j} dx^i dx^j$$

$$= \sum_{1 \le i,j \le 2(n+1)} \left(\frac{\partial^2 K(\boldsymbol{x}, \boldsymbol{x}')}{\partial x^i \partial x'^j} \right)_{\boldsymbol{x}=\boldsymbol{x}'} dx^i dx^j.$$

Therefore, the Riemannian metric induced on S can be written as

$$g_{ij}(\boldsymbol{x}) = \left(\frac{\partial^2 K(\boldsymbol{x}, \boldsymbol{x}')}{\partial x^i \partial x'^j} \right)_{\boldsymbol{x}=\boldsymbol{x}'}.$$

It shows how a local area in I is magnified (by the factor $g_{ij}(\boldsymbol{x})$) in F under the mapping $\Phi(\boldsymbol{x})$.

Conformal transformation

A conformal transformation preserves both angles and the shapes of infinitesimally small figures, but not necessarily their size. The original idea of Amari and Wu [6] and Wu and Amari [7] of conformal transformation of the kernel function is to enlarge the magnification factor $g_{ij}(\boldsymbol{x})$ around the separating boundary but reduce it around other points far from the boundary by modifying the kernel K as

$$\widetilde{K}(\boldsymbol{x}, \boldsymbol{x}') = D(\boldsymbol{x}) D(\boldsymbol{x}') K(\boldsymbol{x}, \boldsymbol{x}')$$

with a properly defined positive function $D(\boldsymbol{x})$. We have

$$\tilde{g}_{ij}(\boldsymbol{x}) = D(\boldsymbol{x})^2 g_{ij}(\boldsymbol{x}) + D_i'(\boldsymbol{x}) D_j'(\boldsymbol{x}) K(\boldsymbol{x}, \boldsymbol{x}) + 2 D_i'(\boldsymbol{x}) D(\boldsymbol{x}) K_i'(\boldsymbol{x}, \boldsymbol{x})$$

where $K_i'(\boldsymbol{x}, \boldsymbol{x}) = \left. \frac{\partial K(\boldsymbol{x}, \boldsymbol{x}')}{\partial x_i} \right|_{\boldsymbol{x}=\boldsymbol{x}'}$ and $D_i'(\boldsymbol{x}) = \frac{\partial D(\boldsymbol{x})}{\partial x_i}$. In order to increase the soft margin around the separating boundary in F, the factor function $D(\boldsymbol{x})$ should be chosen in a way such that $\tilde{g}_{ij}(\boldsymbol{x})$ has greater values around the separating boundary.

Typical factor functions are suggested by many authors. See for example, Amari and Wu [6], Wu and Amari [7], Wu and Chang [15] and Williams, Li, Feng and Wu [11]. In our study, we adopt Williams, Li, Feng and Wu [11], presenting the form of the factor function as

$$D(\boldsymbol{x}) = e^{-\kappa f(\boldsymbol{x})^2} \tag{12}$$

with the combination of the primary kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2} \right). \tag{13}$$

It should be noted that $f(\boldsymbol{x})$ is given by the decision boundary (2) and takes its maximum of the separating region where $f(\boldsymbol{x}) = 0$, and decays to $e^{-\kappa}$ at the margins of the separating region where $f(\boldsymbol{x}) \pm 1$. One can refer the original idea Williams, Li, Feng and Wu [11] to see why this function form improve the accuracy. Remaining crucial problem is to optimize positive constant κ to improve

classification for the given training data set, but it would be accomplished by iterated calculations. In our empirical study, we will call this modification as “D-modified SVM” to distinguish from primary SVM. The detailed computational algorithm for D-modified SVM is discussed in Kijima and Takada [8].

Empirical feature space

We introduce here the empirical feature space according to Xiong, Swamy and Ahmad [9] and Lin, Jiang, Zhao, Pang and Ma [10]. Let $\{\boldsymbol{x}_k\}_{k=1,\cdots,m}$ be a $2(n+1)$ dimensional training data set and $\boldsymbol{X}$ denote the $m \times 2(n+1)$ sample matrix whose k-th rows consists of $\boldsymbol{x}_k$. By introducing a specific map from the input data space I to an r-dimensional Euclidean space R^r ($r \leq m$), determined bellow, one can pre-optimize the conformal transform in terms of the training data without training SVM. Suppose that $m \times m$ matrix $K = (k_{ij})_{1\leq i,j\leq m}$ with $k_{ij} \equiv \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ has rank r ($r \leq m$). Since K is a symmetric positive semidefinite, it has orthogonal decomposition as $K = P\Lambda P^{\mathsf{T}}$, where Λ is a diagonal matrix consisting of only r positive eigenvalues of K in decreasing order, and P consists of the eigenvectors corresponding to the ordered positive eigenvalues of K. Define the map Φ_r^e from the input data space to an r-dimensional Euclidean space; $\Phi_r^e : I \to R^r$ as

$$\boldsymbol{x} \mapsto \Lambda^{-\frac{1}{2}} P^{\mathsf{T}} \big(K(\boldsymbol{x}, \boldsymbol{x}_1), K(\boldsymbol{x}, \boldsymbol{x}_2), \cdots K(\boldsymbol{x}, \boldsymbol{x}_m)\big)^{\mathsf{T}}.$$

We shall call the embedding space $\Phi_r^e(I)$ as the empirical feature space. As Xiong, Swamy and Ahmad [9] and Lin, Jiang, Zhao, Pang and Ma [10] pointed out, the empirical feature space preserves the geometrical structure, i.e., distances and angles, of $\{\Phi(\boldsymbol{x}_i)\}$ in the feature space. This can be understood by calculating dot product matrix of $\{\Phi_r^e(\boldsymbol{x}_i)\}_{i=1,\cdots,m}$ as

$$\begin{aligned}\Big(\Phi_r^e(\boldsymbol{x}_i) \cdot \Phi_r^e(\boldsymbol{x}_j)\Big)_{1\leq i,j\leq m} &= \Big(\big(\Phi_r^e(\boldsymbol{x}_i)\big)^{\mathsf{T}} \Phi_r^e(\boldsymbol{x}_j)\Big)_{1\leq i,j\leq m} = KP\Lambda^{-\frac{1}{2}} \Big(KP\Lambda^{-\frac{1}{2}}\Big)^{\mathsf{T}} \\ &= KP\Lambda^{-1}P^{\mathsf{T}}K = K = \Big(\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)\Big)_{1\leq i,j\leq m}.\end{aligned}$$

Therefore, the training data has the same class separability in both the empirical feature space and the feature space. Remembering the kernel trick, we see that one cannot control $\Phi(\boldsymbol{x})$ alone to improve the classification performance, but instead can control $\Phi_r^e(\boldsymbol{x})$ because it has $K(\boldsymbol{x}, \boldsymbol{x}_i)$ in its definition, which are accessible element, and doesn’t contain $\Phi(\boldsymbol{x})$ anymore. By virtue of this adroit idea, effective pre-optimization of the data-dependent conformal transform in the empirical feature space can be achieved before solving quadratic programing problem of the SVM.

Data-dependent Optimized kernel

In this subsection, we introduce so-called data-dependently conformal transformed kernel $\widetilde{K}$ according to Xiong, Swamy and Ahmad [9] and Lin, Jiang, Zhao, Pang and Ma [10]. They assumed for factor function as

$$Q(\boldsymbol{x}) = \alpha_0 + \sum_{i=1}^{\ell} \alpha_i k(\boldsymbol{x}, \boldsymbol{a}_i)$$

where $k(\boldsymbol{x}, \boldsymbol{a}_i) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{a}_i\|^2}, \boldsymbol{a}_i \in R^{2(n+1)}$, γ is a free parameter, and α_i's are the combination coefficients. The set $\{\boldsymbol{a}_i; i = 1, \cdots, \ell\}$ is called empirical cores which can be chosen from the training data or determined according to the distribution of the training data. And next introduce a measure for the class separability in the empirical feature space and survey an efficient algorithm for optimization of the combination coefficients. One sees that in matrix notation,

$$\widetilde{K} = \left[Q(\boldsymbol{x}_i)K(\boldsymbol{x}_i, \boldsymbol{x}_j)Q(\boldsymbol{x}_j)\right] = QKQ,$$

where Q is a diagonal matrix, whose diagonal elements are $\{Q(\boldsymbol{x}_1), Q(\boldsymbol{x}_2), \cdots, Q(\boldsymbol{x}_m)\}$. We use the notation for vectors $\boldsymbol{q} = \left(Q(\boldsymbol{x}_1), Q(\boldsymbol{x}_2), \cdots, Q(\boldsymbol{x}_m)\right)^{\mathsf{T}}$ and $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_\ell)^{\mathsf{T}}$. Then we have

$$\boldsymbol{q} = \begin{pmatrix} 1 & k(\boldsymbol{x}_1, \boldsymbol{a}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{a}_\ell) \\ 1 & k(\boldsymbol{x}_2, \boldsymbol{a}_1) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{a}_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & k(\boldsymbol{x}_m \boldsymbol{a}_1) & \cdots & k(\boldsymbol{x}_m, \boldsymbol{a}_\ell) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_\ell \end{pmatrix}$$

As we see the training data has the same class separability in both the empirical feature space and the feature space, and we can control $\Phi_r^e(\boldsymbol{x})$ directly, it is better to measure the class separability in the empirical data space. We rely on the following quantity called Fisher scalar for measuring the class separability in the empirical feature space.

$$J = \frac{\operatorname{tr} S_b}{\operatorname{tr} S_w}$$

Here, S_b is the between-class scatter matrix and S_w is the within-class scatter matrix, defined as follows. Let the number of samples in the first class with label -1 be m_1, and the number of samples in the second class with label +1 be m_2 and set $m = m_1 + m_2$. Let $\boldsymbol{y}_i, i = 1, \cdots, m$ be the images of the training data in the empirical feature space, and $\overline{\boldsymbol{y}}$, $\overline{\boldsymbol{y}}_1$ and $\overline{\boldsymbol{y}}_2$ denote the center of the entire training data, that of class with label -1, and that of class with label +1, respectively. Then two matrices S_b and S_w are defined by

$$S_b = \frac{1}{m}\sum_{i=1}^{2} m_i(\overline{\boldsymbol{y}}_i - \overline{\boldsymbol{y}})(\overline{\boldsymbol{y}}_i - \overline{\boldsymbol{y}})^{\mathsf{T}},$$

$$S_w = \frac{1}{m}\sum_{i=1}^{2}\sum_{j=1}^{m_i}(\boldsymbol{y}_i^j - \overline{\boldsymbol{y}}_i)(\boldsymbol{y}_i^j - \overline{\boldsymbol{y}}_i)^{\mathsf{T}},$$

where the vector $\boldsymbol{y}_i^j$ denotes the jth data in the ith class ($i = 1,2$). Without loss of generality, we can arrange the order of training data so as to decompose the kernel matrix into following block structure, indicating that the first m_1 data are belonging to the first class (label: -1), and the remaining m_2 data are belonging to the second class (label: +1).

$$\widetilde{K} = \begin{pmatrix} \widetilde{K}_{11} & \widetilde{K}_{12} \\ \widetilde{K}_{21} & \widetilde{K}_{22} \end{pmatrix}$$

Block matrices $\widetilde{K}_{11}, \widetilde{K}_{12}, \widetilde{K}_{21}$ and $\widetilde{K}_{22}$ represents the submatrix of $\widetilde{K}$ of size $m_1 \times m_1$, $m_1 \times m_2$, $m_2 \times m_1$ and $m_2 \times m_2$ respectively. Then the "between-class" and "within-class" kernel scatter matrices $\tilde{B}$ and $\widetilde{W}$ are determined respectively as follows.

$$\tilde{B} = \begin{pmatrix} \frac{1}{m_1}\widetilde{K}_{11} & 0 \\ 0 & \frac{1}{m_2}\widetilde{K}_{22} \end{pmatrix} - \begin{pmatrix} \frac{1}{m}\widetilde{K}_{11} & \frac{1}{m}\widetilde{K}_{12} \\ \frac{1}{m}\widetilde{K}_{21} & \frac{1}{m}\widetilde{K}_{22} \end{pmatrix}, \qquad \widetilde{W} = \begin{pmatrix} k_{11} & 0 & \cdots & 0 \\ 0 & k_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_{mm} \end{pmatrix} - \begin{pmatrix} \frac{1}{m_1}\widetilde{K}_{11} & 0 \\ 0 & \frac{1}{m_2}\widetilde{K}_{22} \end{pmatrix}.$$

Similarly, "between-class" and "within-class" kernel scatter matrices B and W corresponding to the kernel K are determined respectively as

$$B = \begin{pmatrix} \frac{1}{m_1}K_{11} & 0 \\ 0 & \frac{1}{m_2}K_{22} \end{pmatrix} - \begin{pmatrix} \frac{1}{m}K_{11} & \frac{1}{m}K_{12} \\ \frac{1}{m}K_{21} & \frac{1}{m}K_{22} \end{pmatrix}, \qquad W = \begin{pmatrix} k_{11} & 0 & \cdots & 0 \\ 0 & k_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_{mm} \end{pmatrix} - \begin{pmatrix} \frac{1}{m_1}K_{11} & 0 \\ 0 & \frac{1}{m_2}K_{22} \end{pmatrix}.$$

As Theorem 1 in Xiong, Swamy and Ahmad [9] asserts, class separability measure J is given by

$$J = \frac{\mathbf{1}_m^{\intercal}\tilde{B}\mathbf{1}_m}{\mathbf{1}_m^{\intercal}\widetilde{W}\mathbf{1}_m} = \frac{\boldsymbol{q}^{\intercal}B\boldsymbol{q}}{\boldsymbol{q}^{\intercal}W\boldsymbol{q}},$$

where $\mathbf{1}_k = (1, \cdots, 1)^{\intercal}$ denotes the k dimensional vector. These representations are useful for numerical calculation of the partial derivative of J with respect to $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_\ell)^{\intercal}$. Let J_1 and J_2 be functions of combination coefficients $\boldsymbol{\alpha}$ respectively as

$$J_1 = J_1\big(q(\boldsymbol{\alpha})\big) = 1_m^{\intercal}\tilde{B}1_m = q^{\intercal}Bq,$$
$$J_2 = J_2\big(q(\boldsymbol{\alpha})\big) = 1_m^{\intercal}\widetilde{W}1_m = q^{\intercal}Wq.$$

From Theorem 2 of Xiong, Swamy and Ahmad [9], we have

$$\frac{\partial J_1}{\partial \alpha_l} = 2K_1^{\intercal}BK_1\alpha_l, \qquad \frac{\partial J_2}{\partial \alpha_l} = 2K_1^{\intercal}WK_1\alpha_l, \qquad (l = 0,1,\cdots,\ell)$$

where

$$\boldsymbol{K_1} = \begin{pmatrix} \mathbf{1} & \boldsymbol{k(x_1, a_1)} & \cdots & \boldsymbol{k(x_1, a_\ell)} \\ \mathbf{1} & \boldsymbol{k(x_2, a_1)} & \cdots & \boldsymbol{k(x_2, a_\ell)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & \boldsymbol{k(x_m a_1)} & \cdots & \boldsymbol{k(x_m, a_\ell)} \end{pmatrix}.$$

Hence the partial derivative of J with respect to α_l $(l = 0,1,\cdots,\ell)$ is then given by

$$\frac{\partial J}{\partial \alpha_l} = \frac{2}{J_2^2}(J_2K_1^{\intercal}BK_1 - J_1K_1^{\intercal}WK_1)\alpha_l = 2\frac{K_1^{\intercal}BK_1 - JK_1^{\intercal}WK_1}{J_2}\alpha_l.$$

Maximization of $J = J(\alpha)$ can be achieved numerically by recursive form as follows.

Kernel Optimization Algorithm (Xiong, Swamy and Ahmad [9])

Step 1. Group the data according to their class labels. Calculate $K, K_1, B,$ and W.

Step 2. Set $\boldsymbol{\alpha}^{(0)} = (1,0,\cdots,0)^{\intercal}$ and $n = 0$.

Step 3. Calculate $\boldsymbol{q} = K_1\boldsymbol{\alpha}^{(n)}$.

Step 4. Calculate $J_2 = \boldsymbol{q}^{\intercal}W\boldsymbol{q}$ and $J = \boldsymbol{q}^{\intercal}B\boldsymbol{q}/\boldsymbol{q}^{\intercal}W\boldsymbol{q}$.

Step 5. Update

$$\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} + \eta(n)\left(\frac{K_1^{\intercal}BK_1 - JK_1^{\intercal}WK_1}{J_2}\right)\boldsymbol{\alpha}^{(n)}$$

and normalize $\boldsymbol{\alpha}^{(n+1)}$ so that $\left\|\boldsymbol{\alpha}^{(n+1)}\right\| = 1$.

Step 6. If n reaches a pre-specified number N, say 200, then stop and quit the loop. Otherwise, set $n = n + 1$ and go to Step 3.

Here, $\eta(n) = \eta_0\left(1 - \frac{n}{N}\right)$ is called the learning rate, which controls convergence speed of the above recursive algorithm. As Xiong, Swamy and Ahmad [9] selected, we set $\eta_0 = 0.01$ in our empirical study. We will call this modification as "Q-modified SVM" to distinguish from D-modified SVM and primary SVM. At the stage of empirical analysis, we have some decision branch for choosing empirical cores $\{\boldsymbol{a}_i; i = 1,\cdots,\ell\}$. For example, Xiong, Swamy and Ahmad [9] suggested to select one third of the training data randomly as the empirical cores. We call this strategy as "Q-modified SVM (1/3)". Alternative way is, along the basic idea of Gang Wu and Edward Chang [15], to use support vectors worked out by calibrating primary SVM. This strategy needs to solve optimization problem twice so time-consuming but recycle of the support vectors would be expected to be efficient. We call this strategy as "Q-modified SVM (sv)" later on.

Numerical Studies

In our empirical study, the training of SVM is refreshed every day (i.e., forget about the yesterday) and we chose the running interval $[T_m, T_{m+k}], (m = 0,1,2,\cdots)$ with fixed constant $k = 60 \times 60$ for training data, where $T_0 =$ 09:00:00 (opening time) and $T_k =$10:00:00 are fixed. Thus, the learning interval for each prediction is set to be a one hour and then we cannot start prediction immediately at the opening time. We note that we had processed the original data into second-scale data and then $T_{m+1} - T_m$ is 1 second.

Here, we must note that the label y_ℓ, which will be apparent at time $t > T_\ell$, associated with training data x_ℓ should be known by the time T_m to take into account for training at T_m. Otherwise one knows the answer before it happens. Therefore, if we are in a time T_m, we can use the data during $[T_{m-k}, T_\ell]$, where $\ell = max\{i \in \{1,2,\cdots,M\} | T_{\mathfrak{T}(T_i)} \le T_m\}$ for learning at T_m. In what follows, we abbreviate the term *training data* $[T_m, T_{m+k}]$ so as to identify the training data and its corresponding time interval. So, if we say *training data* $[T_m, T_{m+k}]$, it indicates the available training data during the time interval $[T_m, T_{m+k}]$. Our simulation procedure with prespecified parameters is summarized as follows.

Step 0. Do nothing during $[T_0, T_k] = [09{:}00{:}00, 10{:}00{:}00]$ and set the current time $T_m = 10{:}00{:}00$.

Step 1. Based on the training data $[T_{m-k}, T_\ell]$, where $\ell = max\{i \in \{1,2,\cdots,M\} | T_{\mathfrak{T}(T_i)} \le T_m\}$, calibrate

the SVM (Primary SVM, D-Modified SVM, Q-Modified SVM(1/3) and Q-Modified SVM(sv)).

Step 2. Predict y_m and store it for each SVM.

Step 3. If $T_m < 15{:}10{:}00$, then set $T_m = T_m + 1$sec. and go to Step 1. Otherwise stop and quit the loop.

We need to determine evaluation method in order to compare the newly proposed prediction models of the modified kernel SVM with the standard one that is treated as a benchmark. Let $n_{1,1}$ be a number that count the event of actual upward direction of the mid-price movements that were correctly predicted as upward direction (true positive), while a number $n_{-1,1}$ count the event of actual upward direction that were incorrectly predicted as down ward direction (false negative). Similarly, let $n_{-1,-1}$ be a number that count the event of actual down ward direction of the mid-price movements that were correctly predicted as down ward direction (true negative), while a number $n_{1,-1}$ count the event of actual down ward direction that were incorrectly predicted as upward direction (false positive). These numbers constitute the confusion matrix, which is often employed in the field of machine learning and specifically the problem of statistical classification. We adopt the precisions both for +1 and -1 defined bellow respectively, as the evaluation measure of performance.

$$\mathbf{Precision}_{(+1)} = \frac{n_{1,1}}{n_{1,1} + n_{1,-1}}, \quad \mathbf{Precision}_{(-1)} = \frac{n_{-1,-1}}{n_{-1,1} + n_{-1,-1}}$$

As we simply focus on the effect of the conformal transformation, we do not involve with the detail of tuning methodology for C and σ. Therefore, we calculate the case with $C \in \{1, 10, 100\}$ and $\sigma = 1$, and then take average of these three cases in terms of precision.

Results and Discussion

The most intrigued results of the empirical analysis would be the following; how much the modified kernel SVM outperforms the primary SVM? The average value of precisions simulated for each methodology are summarized in Table 2. We can see that “Q-Modified SVM (1/3)” and “Q-Modified SVM (sv)” outperform “Primary SVM” slightly both for $\mathbf{Precision}_{(+1)}$ and $\mathbf{Precision}_{(-1)}$ in average between April 2, 2012 and June 30, 2012. We can also see that “Q-Modified (sv)” slightly outperforms “Q-Modified SVM (1/3)”. For the empirical cores, the “Q-Modified (sv)” uses the support vectors, which are located in neighborhood of the decision boundary, so that the modification function $Q(\boldsymbol{x})$ would be efficiently calibrated to given training data rather than “Q-Modified SVM (1/3)”. In particular, when the number of the label +1 and -1 at training stage are imbalance, random sampling may have thrown away some sparse data. The most remarkable results are that “D-Modified SVM” with cautiously chosen parameter κ outperforms “Primary SVM” more than 3.5% in average. Here, as the next Figures 2 and 3 show, we choose $\kappa = 1.6$ both for $\mathbf{Precision}_{(+1)}$ and $\mathbf{Precision}_{(-1)}$ to achieve best performance. Of course, if we select the model parameter κ separately for upward direction (+1) and downward direction (-1), then the performance would slightly improve.

Table 2: $\mathbf{Precision}_{(+1)}$ and $\mathbf{Precision}_{(-1)}$ in average between April 2, 2012 and June 30, 2012

	Primary SVM	Q-Modified SVM (1/3)	Q-Modified SVM (sv)	D-Modified SVM
$\mathbf{Precision}_{(+1)}$	54.340%	54.886%	55.241%	57.803%
$\mathbf{Precision}_{(-1)}$	54.566%	55.471%	56.578%	58.180%

Figures 2 and 3 show how the $\mathbf{Precision}_{(+1)}$ and $\mathbf{Precision}_{(-1)}$ vary with respect to κ for each month. As is clear from (13), $\kappa = 0$ corresponds to the standard SVM and then by viewing these figures we can see how much the precisions improved when we change the value of κ. Although we cannot know the optimal value of κ in advance, from practical point of view, it would be possible to decide appropriate vale of κ empirically. In this sense, we can optimize the conformal transform of the Gaussian kernel.

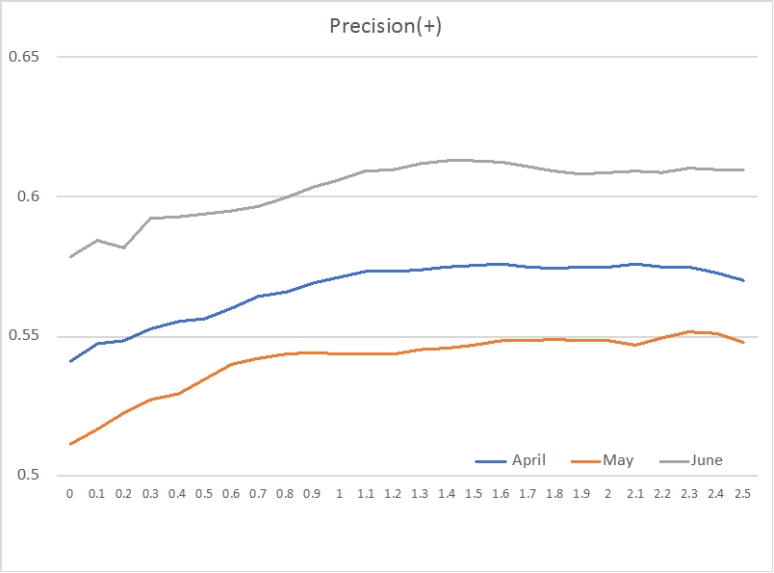

Figure 2: $\mathbf{Precision}_{(+1)}$ for each month. Horizontal axis denotes the value of κ.

To proceed our algorithm in a real market, total computational time of for one prediction should be sufficiency smaller than one tick, i.e., 1 second. In our Matlab computational environment of iMac with 4GHz Intel Core i7, 32GB 1867MHz DDR3, computational time of "D-Modified SVM" for one prediction is 0.176 second in average, while that of "Q-Modified SVM" takes more than 1.48 second. The calculation time depends on the number of empirical cores and this algorithm will not work on the real market without any improvements. As a benchmark, primary SVM takes 0.095 seconds in average for one prediction.

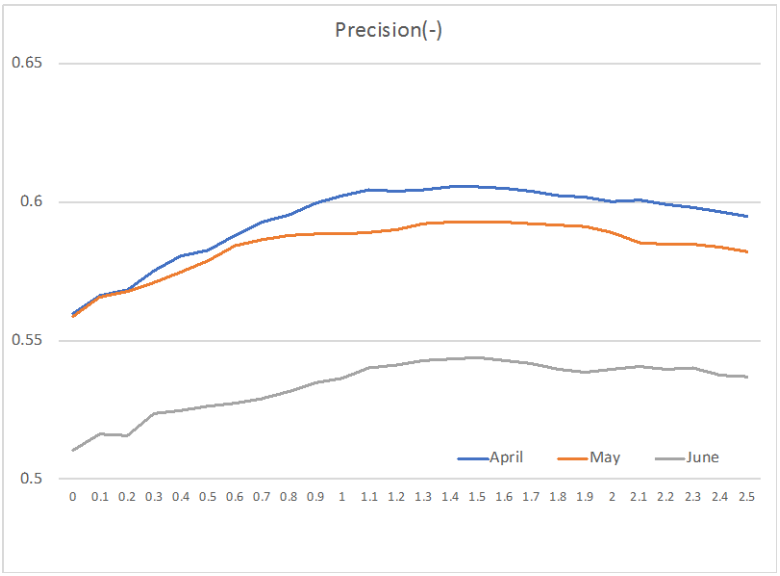

Figure 3: $\mathbf{Precision}_{(-1)}$ for each month. Horizontal axis denotes the value of κ.

Conclusions

As our companion paper Kijima and Takada [8] and others reported, SVM would be a one of the best machine learning technique for predicting mid-price direction by using LOB. In this paper, we found effectiveness of the SVM equipped with conformally transformed Gaussian kernel to improve precision and studied the sensitivity with respect to the parameter κ, which control the degree of the conformal transform. The data-dependent kernel optimization methods are also experimented empirically, however, they failed to outperform Amari and Wu's method I case of the financial high frequency data. For numerical computation, although Amari and Wu's method needs twice as much as the standard Gaussian kernel (for detailed description of computational algorithms, see Kijima and Takada [8]), conformal transform method would be widely applied to high frequency data analysis in finance both from accuracy and computational time.

Acknowledgment

The authors are grateful to participants of the 18th SIG-Financial Informatics conference in Tokyo.

References

[1] Ban Zheng, Eric Moulines and Frederic Abergel (2013) Price Jump Prediction in a Limit Order Book, Journal of Mathematical Finance, Vol. 3, 242-255.

[2] Alec N. Kercheval and Yuan Zhang (2015) Modelling high-frequency limit order book dynamics with

support vector machines, Quantitative Finance, Vol. 15, Issue 8, 1315-1329.
[3] Shangkun Deng, Akito Sakurai and Kei Shioda (2011) Prediction of Foreign Exchange Market States with Support Vector Machine, 10th International Conference on Machine Learning and Applications and Workshops, Vol. 1, 327-332.
[4] Jigar Patel, Sahil Shah, Priyank Thakkar and K Kotecha (2015) Predicting stock and stock price in- dex movement using Trend Deterministic Data Preparation and machine learning techniques, Expert Systems with Applications, Vol 42, 259-268.
[5] Tristan Fletcher and John Shawe-Taylor (2013) Multiple Kernel Learning with Fisher Kernels for High Frequency Currency Prediction, Computational Economics, Vol. 42, Issue 2, 217-240.
[6] Shun-ichi Amari and Si Wu (1999) Improving support vector machine classifiers by modifying kernel functions, Neural Networks, 12, 783-789.
[7] Si Wu and Shun-ichi Amari (2002) Conformal transformation of kernel functions: A data-dependent way to improve the performance of support vector machine classifiers, Neural Processing Letter, 15, 59-67.
[8] Hayato Kijima and Hideyuki Takada (2017) Kernel Modification Effects for Support Vector Machine Applied to Limit Order Book of Nikkei 225 Futures, Journal of Mathematical and Statistical Science, Vol. 2017, Issue 6, 149-167.
[9] Huilin Xiong, M. H. S. Swamy and M. Omair Ahmad (2005) Optimizing the Kernel in the Empirical Feature Space, IEEE Transactions on Neural Networks, Vol. 16, No. 2, 460-474.
[10] Chuang Lin, Jifeng Jiang, Xuefeng Zhao, Meng Pang and Yanchun Ma (2015) Supervised Kernel Optimized Locality Preserving Projection with Its Application to Face Recognition and Palm Biometrics, Mathematical Problems in Engineering, Vol. 2015, Article ID 421671, 10pages, Hindawi Publishing Corporation.
[11] Peter Williams, Sheng Li, Jianfeng Feng and Si Wu (2005) Scaling the kernel function to improve performance of the Support Vector Machine Advances in Neural Networks, Volume 3496 of the series Lecture Notes in Computer Science, 831-836.
[12] Rama Cont, Sasha Stoikov and Rishi Talreja (2010) A stochastic model for order book dynamics, Operations Research, Vol. 58, No. 3, May-June 2010, 549-563.
[13] Martin D. Gould, Mason A. Porter, Stacy Williams, Mark Macdonald, Daniel J. Fenn and Sam D. Howison (2013) Limit order books, Quantitative Finance, Vol. 13, No. 11, 1709-1742
[14] Gerry Tsoukalas, Jiang Wang, Kay Giesecke (2017) Dynamic Portfolio Execution, Management science, Published online in Articles in Advance 27 Oct 2017.
[15] Gang Wu and Edward Y. Chang (2003) Adaptive Feature-Space Conformal Transformation for Imbalanced-Data Leraning, Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003), Washington DC.

Published: Volume 2018, Issue 8 / August 25, 2018