A Survey of Time Series Prediction Using SVM

Yongning Ma

October, 1 2012

Abstract

This expository paper is a result of the reading project for our knowledge engineering class. Our task is to read a survey on time series prediction [20] and identify the key ideas of the paper. Based on our understanding, we give a summary of the theory and applications of time-series prediction using SVM, with a more detailed introduction to bridge the knowledge gap in theoretical background. By reviewing literature, we conclude that SVM provides reliable, accurate and efficient estimation of time series data in a wide array of industries, that often outperforms traditional techniques such as neural network and autoregressive models.

Contents

1 Introduction 2

2 Time Series Prediction 2
  2.1 Background and notations .............................. 2
  2.2 Autoregressive Model ............................... 3

3 Support Vector Machine 5
  3.1 The Vapnik-Chervonenkis Theory .................... 5
  3.2 Linear Support Vector Machines ...................... 6
  3.3 Support Vector Machines with Kernels ............... 8
  3.4 Variations and Hybrid Systems ...................... 9

4 Industrial Applications 10
  4.1 Financial Market Forecast ......................... 10
  4.2 General Business Application ...................... 10
  4.3 Environmental Parameter Estimation ................ 11

5 Conclusion 11

1We wish to thank Dr. Ding for her guidance and support.
1 Introduction

Time series prediction is ubiquitous and plays critical role in many fields, such as financial market prediction, environment state forecasting and business analytics. However the prediction of time series is very difficult task: it asks the system to forecast the behavior of a complex system, while only simplistic data points along the time axis are given. The data can only be treated as stochastic in nature; furthermore, we can’t assume any a-priori structure of the data (for example, the data points can no longer be assumed as independent of each other; in time-series prediction the present is often determined by the past). For this reason, the very mature and efficient linear techniques often fail to give satisfactory results.

On the other hand, SVM technology provides good performance for non-linear problems, and at the same time do not require prior knowledge of the structure of the data. This makes SVM an ideal choice for time series prediction. Another comparable technology is neural network, which also solves non-linear regression problems non-parametrically, but converge differently, and is less efficient.

We attempt to give a comprehensive survey on time series prediction using SVM. The survey contains an introduction to time series prediction problems, a prime on SVM and its application to time series prediction, and finally the performance of SVM based systems in real-life applications.

A side-note: the topic crosses different fields, each with a set of slightly different terminologies of its own flavor. We try to use consistent conventions throughout this note, with some sacrifice of brevity. Our presentation style follows that of [11], with a few modifications: 1) We use bold font for vectors throughout the text. 2) To distinguish the indexes in a dataset from that in a time series, we use a superscript to denote the $i$-th training data e.g. $(y^i, x^i)$ while the subscript denotes the $t$-th sample in a time series e.g. $y_t$, or the $i$-th component of a vector, e.g. $x_i$.

2 Time Series Prediction

2.1 Background and notations

The study of time-series prediction is motivated by many important applications in various fields, such as econometrics, mathematical finance, dynamic systems and signal processing. [9] is a classic treatment of the field; our development of time series prediction model borrows the more contemporary conventions from [10]. For readability, it is useful to define a few notions as follows.

A time series can be described by a sequence of observational data over time:

$$\{y_t(\omega)\}, \ t = 1, 2, \ldots, T$$

Conceptually, a time series the realization (or trajectory) of a stochastic process $\{y_t(\omega)\}_{t \in \mathbb{T}}$ where $\mathbb{T}$ is a suitable index set. A strictly stationary process is a process for which the joint probability distribution of $(y_t(\omega), y_{t-1}(\omega), ..., y_{t-k}(\omega))$
is independent of \( t \forall k \). A univariate time series is given by \( y_t: \Omega \rightarrow \mathbb{R} \) that maps an event \( \omega \) to a scalar \( y_t(\omega) \) while \( y_t: \Omega \rightarrow \mathbb{R}^n \) defines a multivariate time series. We limit our scope to univariate time series in this note.

The task of time series prediction is to forecast the value of \( y_t(\omega) \) at time \( t \) based on a set of past data \( \{y_s(\omega)\}_{s<t} \) by a function \( f \)

\[
y_t(\omega) = f(\{y_s(\omega)\}_{s<t}) \tag{1}
\]

Practically only finite number of samples can be generated from the past, and \( f \) is estimated by \( \hat{f}: \mathbb{R}^n \rightarrow \mathbb{R} \):

\[
\hat{y}_t(\omega) = \hat{f}(x_t) \tag{2}
\]

In machine learning terms, the task of predicting is to learn the mapping \( f \) from a given training dataset \( D = \{(y_1^t, x_1^t), (y_2^t, x_2^t), \ldots, (y_n^t, x_n^t)\} \subset \mathbb{R} \times \mathcal{X} \), where \( y^t \) is the output variable, or labels that correspond to a “truth value” given the input \( x^t = (y_{t-n}, y_{t-n-1}, \ldots, y_{t-1}) \), an \( n \)-dimensional vector representing a set of past data. The objective of the learning algorithm is to find a prediction \( \hat{f} \) that minimizing total risk \( R_{\text{total}} = \mathbb{E}[L(y_t, \hat{f}(x_t))], \) formally

\[
\hat{f} = \arg\min_{f} \int_{\mathcal{X} \times \mathcal{Y}} L(y_t, \hat{f}(x_t))dP(x_t, y_t) \tag{3}
\]

where \( L(y_t, \hat{f}(x_t)) \) is the loss function that describes the degree of penalty associated with a wrong prediction. The sum of squared residual is by far the most common:

\[
L(y_t, \hat{f}(x_t)) = \text{RSS}(\hat{f}) = \sum_{i=1}^{n}(y_i - \hat{f}(x_t))^2 \tag{4}
\]

Common loss functions are listed in Table 12.1 in [11] and Table 1 in [22].

Function \( \hat{f} \) can be expressed as \( \hat{\theta}, \delta \), where \( \hat{\theta} \) is a set of parameters and \( \delta \) is a association rule, or model. In practice \( \delta \) is often fixed and we approximate \( f \) by determining \( \hat{\theta} \). The best known model for predicting time series, ARIMA (Autoregressive Integrated Moving Average), described in [5], is due to Box, G. E.. Another useful statistical model is HMM (Hidden Markov Model), and a well-known example is the Kalman filter [14].

### 2.2 Autoregressive Model

The widely-used autoregressive model is of the form

\[
y_t(\omega) = c + \langle \beta, x_t \rangle + \epsilon_t \tag{5}
\]

where \( \beta = (\beta_1, \beta_2, \ldots, \beta_n) \) is the coefficient to be determined and \( \epsilon_t = y_t - \hat{y}_t \) is the white noise with \( \mathbb{E}(\epsilon_t) \overset{\Delta}{=} 0 \). In the AR(1) process (when \( n = 1 \)), the model correspond to a linear model of the Markov process, where the future value is only dependent on the present value.
There are very mature techniques to solve the autoregression problem. The simplest is the ordinary least square estimation:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

(6)

where \( X = (x_{t1}, x_{t2}, ..., x_{tn})^T \) and \( y = (y_1, y_2, ..., y_n)^T \). The autoregressive model works on the assumption that the process is stationary and ergodic, thus a Wold decomposition is possible and \( \hat{\beta} \rightarrow^p \beta \). For the simple case of \( AR(1) \) process, stationarity requires \( |\beta| < 1 \). However for \( AR(n) \) process the behavior is much more complex.

To generalize the AR model in order to capture the structure of the random process, we can consider the ARMA (autoregressive moving-average) model

$$y_t(\omega) = c + \langle \beta, x_t \rangle + \langle \theta, \varepsilon \rangle + \varepsilon_t$$

(7)

In the ARMA model the white noise is broken into a linear combination of \( p \) white noises, where \( \varepsilon = (\varepsilon_{t-p-1}, \varepsilon_{t-p}, ..., \varepsilon_{t-1}) \) and \( \varepsilon_t \) is independently distributed white noise.

If the process is non-stationary, it would be suitable to use ARIMA (autoregressive integrated moving-average) model to capture the non-stationary information. First introduce the difference operator \( \Delta \triangleq 1 - L \) where \( L \) is the lag operator with \( Lx_n = x_{n-1} \). It becomes clear that the difference operator reduces an increasing trend \( x_t = at + b \) to a constant \( b \) and more generally, \( \Delta^d \) reduces polynomial of degree \( d \) to constant, therefore helps to break-down a non-stationary process to the product of a stationary process and a non-stationary process:

$$(\Delta^d y_t(\omega)) = c + \langle \beta, x_t \rangle (\Delta^d y_t(\omega)) + \langle \theta, \varepsilon \rangle + \varepsilon_t$$

(8)

In an ARIMA(p,d,q) process \( \beta \in \mathbb{R}^{p-d} \) and \( \theta \in \mathbb{R}^q \). The ARIMA model is very sophisticated, but still has limited capability to deal with non-linear process. To extend the model to non-linear case, replace \( x_t \) in (5) with some function \( \phi : \mathbb{R}^n \rightarrow \mathbb{F} \) (not necessarily linear) that transforms our sample space (or input space, \( \mathbb{R}^n \)) into a feature space \( \mathbb{F} \). If the function \( \phi \) successfully maps the input samples into linearly separable features, the general problem in the sample space \( \mathbb{R}^n \) is equivalent to a linear regression problem in \( \mathbb{F} \) and can be directly solved by equation (6):

$$y_t(\omega) = c + \langle \beta, \phi(x_t) \rangle + \varepsilon_t$$

(9)

Note that \( \beta \in \mathbb{F} \). Unfortunately the dimension of the feature space can be very high and the computation of the inner product \( \langle \beta, \phi(x_t) \rangle \) is quite often intractable ("the curse of dimensionality" [3]). To develop a practical solution,

---

2The Wold’s theorem states that any stationary, ergodic process can be expressed as the sum of a deterministic process (therefore convergence is promised) plus a linear combination of white noise. It provides theoretical background for ARMA and ARIMA model as well. In a separate note, the ergodic theorem states that for an ergodic and stationary process, \( \frac{1}{n} \sum_{t=1}^{n} y_t \rightarrow^a \mathbb{E}(y) \), a result similar to the law of large numbers that ensures the consistency of an estimation.
we proceed to the next section, where the theory of support vector machine provides a elegant way to tackle the problem. (It is sometimes called SVR, support vector regression when used for prediction rather than classification). [8] and [21] provides comprehensive introduction to SVM and kernel machines. Two great additional source of general information on SVM are [7] and [22]. A general framework for predicting time-series using SVM is described in [18].

3 Support Vector Machine

3.1 The Vapnik-Chervonenkis Theory

We start by considering a common linear regression problem (5). The SVM follows structural risk minimization (SRM) principal in Vapnik-Chervonenkis (VC) theory, developed by Vladimir Vapnik and Alexey Chervonenkis and described in [26], [27] and [28]. SRM suggests a optimal learning algorithm chooses a \( \hat{f} \) to minimize the total risk (or real risk)

\[
R_{\text{total}}(\hat{f}) = \int_{X \times Y} L(y_t, \hat{f}(x_t)) dP(x_t, y_t)
\]

(10)

Since \( P(x_t, y_t) \) is not known a priori, \( R(\hat{f}) \) can’t be estimated directly. Instead, the empirical risk is fixed for a specific training set \( D \) and can be computed as follows:

\[
R_{\text{emp}}(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} L(y_t, \hat{f}(x_t))
\]

(11)

where \( L \) is a suitable loss function. Then Vapnik proved in [29] the following bound holds with a probability of \( 1 - \eta \):

\[
R_{\text{total}}(\hat{f}) \leq R_{\text{emp}}(\hat{f}) + \sqrt{\frac{h \log(\frac{2h}{\eta}) + \log \left( \frac{2}{\eta} \right)}{l}}
\]

(12)

where \( h \) is an integer called the VC-dimension, a measure of the capacity (or complexity) of the function \( \hat{f} \). The more complex a machine is, the better it can deal with the patterns appearing in the training set and conversely, the worse it can be generalized to larger data samples. It is a surprising result that the bound is not related to the probability distribution \( P(x, y) \) but rather VC-dimension. However the VC dimension is not directly computable. In relative to our linear regression problem, it’s convenient that the following bound is also proved for the VC-dimension of the linear classifier \( G(x) = \text{sign}[x^T \beta + c] \) with \( ||\beta|| \leq A \):

\[
h < R^2 A^2
\]

(13)

where \( R \) is the smallest radius of a sphere that contains all the training data. Hence minimizing \( ||\beta|| \) reduces the complexity of the model and increases its generalization power. Interestingly enough, \( 1/||\beta|| \) also correspond to the ”margin” of the separating hyperplanes.
3.2 Linear Support Vector Machines

Referring back to the regression problem in (5) with dataset $D$ as defined previously. The optimization problem

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \beta \| \\
\text{subject to} & \quad y_i - c - \langle \beta, x_i \rangle \leq \epsilon \\
& \quad c + \langle \beta, x_i \rangle - y_i \leq \epsilon
\end{align*}$$

for all $(y_i, x_i)$, $i$ being the index of the training dataset, finds the optimal $\hat{f}$ on the assumption that $\hat{f}$ approximates all data sets with $\epsilon$ precision. In practice (14) not always feasible, and a relaxation is introduced to the constraints via slack variables $\xi^i$ with a suitable constant $C$ that controls the balance between the contradicting goals of minimizing $\| \beta \|$ and fitting data to the regression line.

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \beta \| + C \sum_{i=1}^{n} \xi^i \\
\text{subject to} & \quad y_i - c - \langle \beta, x_i \rangle \leq \epsilon + \xi^i \\
& \quad c + \langle \beta, x_i \rangle - y_i \leq \epsilon + \xi^i
\end{align*}$$

The slack variables can be interpreted as a loss function. Letting $r = |y_i - c - \langle \beta, x_i \rangle|$, define the $\epsilon$-intensive loss function (or penalty function)

$$L(y_i, \hat{f}(x_i)) = \begin{cases} 
0 & \text{if } r < \epsilon, \\
 r - \epsilon & \text{otherwise}
\end{cases}$$

With the loss function so defined, the constraints are now always satisfied. Another choice of the loss function would be the squared error loss function:

$$L(y_i, \hat{f}(x_i)) = r^2$$

The squared error loss function is used in constructing the least-square SVM. We shall see that such choice of loss function has advantage in terms of computation complexity.

Convex programming technique is used to solve (15). Details about convex optimization problems can be found in [6]. The solution outline is: first compute the prime Lagrangian $L_P$ and set the partial derivative of the Lagrangian to zero, then substitute the resulted equations back to Lagrangian to get a dual $L_D$. The original optimization problem is equivalent to maximize $L_D$ with the variables $\alpha^i$, the Lagrangian multiplier.

First compute the Lagrangian for problem (15):
\[ L_P = \frac{1}{2} \| \beta \| + C \sum_{i=1}^{n} \xi^i - \sum_{i=1}^{n} \mu^i \xi^i \]
\[ \quad - \sum_{i=1}^{n} \alpha^i_1 (\epsilon + \xi^i - y^i_t + c + \langle \beta, x^i_t \rangle) \]
\[ \quad - \sum_{i=1}^{n} \alpha^i_2 (\epsilon + \xi^i + y^i_t - c - \langle \beta, x^i_t \rangle) \]  
(18)

By the saddle point condition, the partial derivatives with respect to variables \( c, \beta, \xi^i \) vanish simultaneously for all \( i \):

\[ \partial_c L_P = \sum_{i=1}^{n} (\alpha^i_1 - \alpha^i_2) = 0 \]  
(19)

\[ \partial_\beta L_P = \beta - \sum_{i=1}^{n} (\alpha^i_1 - \alpha^i_2) x^i_t = 0 \]  
(20)

\[ \partial_\xi L_P = C - \alpha^i_1 - \alpha^i_2 - \mu = 0 \]  
(21)

Substituting (19) - (21) back to (18), the dual Lagrangian is

\[ L_D = - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha^i_1 - \alpha^i_2)(\alpha^j_1 - \alpha^j_2) \langle x^i_t, x^j_t \rangle \]
\[ \quad - \sum_{i=1}^{n} y^i_t (\alpha^i_1 - \alpha^i_2) \]
(22)

The dual optimization problem is

\[
\text{maximize} \quad \alpha^i_1, \alpha^i_2 \quad - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha^i_1 - \alpha^i_2)(\alpha^j_1 - \alpha^j_2) \langle x^i_t, x^j_t \rangle \\
\quad - \sum_{i=1}^{n} y^i_t (\alpha^i_1 - \alpha^i_2) 
\]

subject to \( \alpha^i_1, \alpha^i_2 \in [0, C] \)

(23)

Though it still looks a daunting problem, the Lagrangian multiplier \( \mu \) has been eliminated in (23) and from (20) it follows that

\[ \beta = \sum_{i=1}^{n} (\alpha^i_1 - \alpha^i_2) x^i_t \]
(24)

\[ \hat{f}(x) = \sum_{i=1}^{n} (\alpha^i_1 - \alpha^i_2) \langle x^i_t, x \rangle + c \]
(25)
(25) is of important use when extending the linear support vector machine to the non-linear case. Furthermore, the Karush-Kuhn-Tucker conditions specify

\[ \alpha^1_i (\epsilon + \xi^1_i - y^i_t + \langle \beta, x^i_t \rangle) = 0 \] (26)
\[ \alpha^2_i (\epsilon + \xi^2_i + y^i_t - c - \langle \beta, x^i_t \rangle) = 0 \] (27)
\[ C - \alpha^1_i - \alpha^2_i = 0 \] (28)

For \( i = 1, 2, \ldots, n \). Together, equations (19)-(28) characterize a uniquely solution to the problem.

3.3 Support Vector Machines with Kernels

As discussed in Section 2, the time-series data is often not linearly separable, and a function \( \phi : \mathbb{R}^n \rightarrow \mathcal{F} \) is used to transform the input space to a feature space, which gives rise to the regression problem in (9). The solution of (9) is given by substituting \( x \) with \( \phi(x) \) in (25)

\[ \hat{f}(x) = \sum_{i=1}^{n} (\alpha^1_i - \alpha^2_i) \langle \phi(x), \phi(x') \rangle + c \] (29)

Recall that the main difficulty in this approach is that the transformed space may have very high or even infinite dimension, making the computation of the solution (29) intractable. It is desirable if a function \( K(, \) , called a kernel function can be found, that allows us to calculate the inner products in \( \mathcal{F} \) directly

\[ K(x, x') = \phi(x) \phi(x') \] (30)

The technique is known as the kernel trick originally proposed in [1] and suggested by Vapnik in [4]. With the kernel function available, it is much easier to calculate

\[ \hat{f}(x) = \sum_{i=1}^{n} (\alpha^1_i - \alpha^2_i) K(x, x') + c \] (31)

In practice though, it is often a reverse-engineering process: \( \phi(x) \) is often not known, and we start by specifying a kernel function and hopefully the kernel function corresponds to a inner product space in which the problem is linearly separable. If the result is disappointing, try different kernels and re-experiment.

In the rest of the section we give an example to show the mechanism of the kernel trick, followed by a condition for which the kernel function can be expressed as the inner product of two vectors.

Consider the quadratic kernel (the subscript indexes the components in a vector)

\[ K(x, x') = \langle x, x' \rangle^2 \] (32)
Expanding terms, we have
\[
K(x, x') = \left( \sum_{i=1}^{n} x_i x'_i \right)^2
\]
\[
= \sum_{i=1}^{n} (x_i x'_i)^2 + \sum_{i=2}^{n} \sum_{j=1}^{i-1} \sqrt{2} x_i x'_i \sqrt{2} x_j x'_j
\quad \text{(33)}
\]

Noticing that the equation is symmetrical with regard to \(x_i\) and \(x'_i\), we find the corresponding transforming function
\[
\phi(x) = (x^2_1, x^2_{n-1}, ..., x^2_1, \sqrt{2} x_n x_{n-1}, ..., \sqrt{2} x_2 x_1)
\quad \text{\((n+1)\) terms} \quad \text{(34)}
\]

Verifying, it is indeed true that we have found a \(\phi : \mathbb{R}^n \to \mathbb{R}^{(n+1)}\) such that
\[
K(x, x') = \langle \phi(x), \phi(x') \rangle = \langle x, x' \rangle^2 \quad \text{(35)}
\]

Finally, the kernel function must satisfy the Mercer’s Condition \(^{3}\). A function \(K(x, y)\) can only be expressed by the inner product of two functions in a feature space if and only if for any \(g(x)\) satisfying
\[
\int g(x)^2 \, dx \text{ is finite} \quad \text{(36)}
\]
the following condition is satisfied:
\[
\int K(x, y)g(x)g(y) \, dx \, dy \geq 0 \quad \text{(37)}
\]

3.4 Variations and Hybrid Systems

The SVM technology provides a flexible framework for creating different solutions based on actual needs. For example, if the \(\epsilon\)-intensive loss function (16) is replaced with squared error loss function (17), we get a least-square SVM, developed by Suykens J.A.K., Vandewalle J. in [24] and [23]. The beauty of the LS-SVM is that after taking partial derivatives of the Lagrangian, the resulting conditions for optimality cancels \(\beta\) and \(r\) from the equations, producing a linear system rather than a quadratic programming problem, hence greatly speeds up the calculation.

Another improvement that can be considered is to use other technologies to optimize the constants in SVM, such as \(C\) and \(\epsilon\). An example of such systems is Recurrent Support Vector Machine with Genetic Algorithms (RVMGA), proposed by Pai and Hong ([13] and [19]). The Genetic Algorithms (GA) were used

\(^{3}\)The Mercer’s condition follows Mercer’s Theorem, presented in [16]. For a more detailed discussion of Mercer’s theorem and feature mapping of commonly used kernels, refer to [17]
to optimize the SVM constants, plus the Gaussian kernel parameters. The recurrent SVM (RSVM) suggests the use of MLP with back propagation combined with the SVM architecture.

As SVM produces crisp decision boundaries, fuzzy technology has been introduced to provide support for cases where the decision boundary could be unclear. Bao et al. in [2] proposed a Fuzzy Support Vector Machines Regression (FSVMR) for predicting the stock composite index of the Shanghai Stock Exchange. The approach assumes that the input is noisy, and a degree of uncertainty was introduced.

Last but not least, it is possible to combine ARIMA model with SVM, producing a model that handles both linear and non-linear data. He et al. in [12] proposed a hybrid algorithm for short term electrical load forecasting that uses the ARIMA model to estimate the linear portion of the electrical load time series data and an SVM to estimate the nonlinear residual, where the residual is the difference between the load data and the linear estimation.

4 Industrial Applications

4.1 Financial Market Forecast

One of the most studied applications of SVM is financial data prediction. Apart from the apparent incentives, it is also technically appropriate to use SVM in such application because the financial data is chaotic in nature and among the most difficult to model. Furthermore, there seems to be very strong link between the past and the future in financial time series. It should not be surprising that SVM demonstrates superior performance relative to other traditional, linear technologies.

SVM technology is used with NN and fuzzy technology to give prediction for S&P500 and several foreign bond indices by Tay and Cao in [25]. The SVM significantly outperformed the BP-NN because of its ability to appropriately fit the data. At the same time, a hybrid system of using SOM (self-organizing map) combined with SVM yielded not only better prediction performance but also superior convergence speed.

4.2 General Business Application

Many have explored the application of SVM-based time-series prediction in the business setting, and the result has been generally very good. Successful applications include electricity price forecasting, credit rating analysis, predicting customer “Churning” rates in auto insurance market, analysis of financial failure of dotcoms and production value prediction of the Taiwanese machinery industry in a study by Pai and Li [19].
4.3 Environmental Parameter Estimation

SVR has also been used for the prediction of environmental parameters such as air quality parameters rainfall estimation and detection, and weather forecasting. For example, Lu et al. in [15] proposed the use of SVM to forecast air quality parameters. The prediction of short-term air quality is typically non-linear. The input data was respirable suspend particles (RSP) as collected with other major pollutants such as nitrogen oxides, etc. Sequential Minimization Optimization (SMO) is used for training and the SVM with a Gaussian kernel was employed, and the loss function used is mean-squared error (MSE). The SVM outperformed RBF-based neural-network by a significant margin, based on the performance in predicting one week ahead. A sensitivity analysis was provided for the free parameters (regularization constant, kernel constants, etc.) and there was no set heuristic for determining these parameters.

5 Conclusion

SVM technology provide us a viable way to predict future values in a time-series. It relatively cheaply solves problem in higher dimension and deals with non-linear case much better than traditional technologies. Since time-series has many important applications, it is our believe that SVM technology holds an important place in today’s data-intensive life. Due to the time constraint, we have left many wonderful applications untouched – of course it’s impossible to cover everything in a single note. We hope to continue to explore this field with great interest.

References


