Appendix A
Derivation of Eq. (3.3)

This appendix presents a derivation of Eq. (3.3) that was used in Chap. 3.

When \( N = 0 \), Eq. (3.3) equals zero. When \( N \) approaches infinity, Eq. (3.3) should also approaches infinity.

Define:

\[
L = \frac{E[X_{N+1}]}{E[X_N]} = \frac{\sum_{i=1}^{N+1} \frac{w_i}{\prod_{k=i}^{N+1}(1-\gamma_k)}}{\sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N}(1-\gamma_k)}} \quad (A.1)
\]

\[
= \frac{1}{1-\gamma_{N+1}} \cdot \left[ \sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N}(1-\gamma_k)} + \frac{w_{N+1}}{\prod_{k=i}^{N+1}(1-\gamma_k)} \right] \quad (A.2)
\]

\[
= \frac{1}{1-\gamma_{N+1}} \left[ 1 + \frac{w_{N+1}}{\sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N}(1-\gamma_k)}} \right] \quad (A.3)
\]

\[
> \frac{1}{1-\gamma_{N+1}} \quad (A.4)
\]

\[
> 1. \quad (A.5)
\]

Since Eq. (3.3) monotonically increases as \( N \) increases with a rate greater than 1, therefore, Eq. (3.3) diverges with \( N \).
Manufacturing process restarts from the first task

![Diagram showing a manufacturing process for a 3-stage single-part product](image)

**Fig. A.1** Manufacturing process for a 3-stage single-part product

An illustration of Fig. 3.2 with three sequential tasks is shown in Fig. A.1. Variables $X_1$, $X_2$, and $X_3$ represent the time to complete all three tasks, the last two tasks, the last task, respectively. They are related by the following equations:

\[
E[X_3] = w_3 + \gamma_3 \cdot E[X_1] + (1 - \gamma_3) \cdot 0 \tag{A.6}
\]

\[
E[X_2] = w_2 + \gamma_2 \cdot E[X_1] + (1 - \gamma_2) \cdot E[X_3] \tag{A.7}
\]

\[
E[X_1] = w_1 + \gamma_1 \cdot E[X_1] + (1 - \gamma_1) \cdot E[X_2] \tag{A.8}
\]

by solving Eqs. (A.6)–(A.8), we can obtain:

\[
E[X_1] = \frac{w_1}{(1 - \gamma_1)(1 - \gamma_2)(1 - \gamma_3)} + \frac{w_2}{(1 - \gamma_2)(1 - \gamma_3)} + \frac{w_3}{1 - \gamma_3} \tag{A.9}
\]

\[
= \sum_{i=1}^{3} \frac{w_i}{\prod_{k=i}^{3} (1 - \gamma_k)}. \tag{A.10}
\]

If the number of stages is extended to $N$, by replacing 3 by $N$ in Eq. A.10, we can get the same expression as Eq. (3.3), which is repeated here:

\[
E[X] = \sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N} (1 - \gamma_k)}. \tag{A.11}
\]

Therefore, Eq. (3.3) holds true for all the values of $N$.

Equation (3.3) can be validated by mathematical induction as follows:

In the base case where $N = 1$, according to the definition of expectation, the expectation, $E[X_1]$, is defined by [1]:

\[
E[X_1] = \sum_{i} x_i p_X(x_i) \tag{A.12}
\]
A Derivation of Eq. (3.3)

Fig. A.2 Manufacturing process for a \((N + 1)\)-stage single-part product

Manufacturing process restarts from the first task

\[
\begin{array}{c}
\text{task } 1 \sim N \\
v_{1 \sim N}
\end{array} \quad \begin{array}{c}
\text{task } N + 1 \\
v_{N + 1}
\end{array} \quad \begin{array}{c}
\text{failure } \gamma_{N + 1} \\
\text{success } 1 - \gamma_{N + 1}
\end{array}
\]

\[X_N \quad w_{N + 1}\]

where \(x_i\) is given by Eq. (B.3) and \(p_X(x_i)\) is given by Eq. (B.7). Therefore, Eq. (3.3) is correct for \(N = 1\).

Assume Eq. (3.3) to be true for some \(N \geq 2\). If one more stage is added at the end of the chain, then there are \(N + 1\) stages. Figure A.2 shows the case when there are \(N + 1\) sequential stages. The first \(N\) stages are simplified as one stage. Variables \(X_N\) and \(X_{N + 1}\) represent the time to complete the first \(N\) tasks and the time to complete all \(N + 1\) tasks, respectively. They are related as follows:

\[
E[X_{N + 1}] = E[X_N] + w_{N + 1} + (1 - \gamma_{N + 1}) \cdot 0 + \gamma_{N + 1} \cdot E[X_{N + 1}] \quad (A.16)
\]

by plugging in Eq. (3.3) into Eq. (A.17), we get:

\[
E[X_{N + 1}] = \sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N+1} (1 - \gamma_k)} + \frac{w_{N + 1}}{1 - \gamma_{N + 1}} \quad (A.18)
\]

Therefore, if Eq. (3.3) is true for some \(N \geq 2\), it is also true for \(N + 1\). Since it has been proved to be true for \(N = 1\), by mathematical induction, Eq. (3.3) holds true for all the values of \(N\).
Appendix B
Derivation of the PMF of Random Variable X

In this appendix, we present a derivation of the probability mass function (PMF) of random variable X described in Sect. 3.2.2.

Random variable \( Q_i \) is the number of failures at task \( v_i \). The value of \( Q_i \) is denoted by \( q_i \). Failures are independent events. The joint probability of the event that \( Q_1 = q_1, \ldots, \) and \( Q_N = q_N \) is computed by Eq. (B.1):

\[
P(Q_1 = q_1, \ldots, Q_N = q_N) = \left( \frac{\sum_{i=1}^{N} q_i}{i!} \right)^N \prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1-\gamma_i)^{N-q_i} \tag{B.1}
\]

where \( \gamma_i \) is the failure probability of task \( v_i \) (defined in Eq. (3.2)).

Discrete random variable \( X \) is the execution time of a chain of sequential tasks. The value of \( X \), which is denoted by \( x \), depends on the values of \( Q_1 \) to \( Q_N \) as follows:

\[
X = \sum_{i=1}^{N} \left( \sum_{k=i}^{N} Q_k + 1 \right) \times w_i \tag{B.2}
\]

where \( w_i \) is the execution time of task \( v_i \) (defined in Eq. (3.1)).

Therefore, a combination of values \((q_1, \ldots, q_N)\) for variables \((Q_1, \ldots, Q_N)\) can result in a value \( x \) for variable \( X \). Note that some combinations of \((q_1, \ldots, q_N)\) can result in the same value of \( X \). Set \( \Phi_{X=x} \) contains all combinations such that they all result in the same \( x \) for \( X \). Therefore, value \( x \) is computed as follows:

\[
x = \sum_{i=1}^{N} \left( \sum_{k=i}^{N} q_k + 1 \right) \times w_i , \quad (q_1, \ldots, q_N) \in \Phi_{X=x}. \tag{B.3}
\]
The probability of the event that \( X = x \) equals the sum of the probabilities of events \( Q_1 = q_1, \ldots, Q_N = q_N \), for all \((q_1, \ldots, q_N) \in \Phi_{X=x}\). The PMF of \( X \) can be expressed as follows:

\[
p_X(x) = P(X = x) = \sum_{(q_1, \ldots, q_N) \in \Phi_{X=x}} P(Q_1 = q_1, \ldots, Q_N = q_N). \tag{B.4}
\]

When \( N \geq 2 \), by plugging in Eq. (B.1), Eq. (B.5) can be written as:

\[
p_X(x) = \sum_{(q_1, \ldots, q_N) \in \Phi_{X=x}} \left[ \frac{\prod_{i=1}^{N} q_i!}{\prod_{i=1}^{N} q_i!} \cdot \prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{\left(\sum_{k=i+1}^{N} q_k + 1\right)} \right]. \tag{B.6}
\]

whereas when \( N = 1 \), Eq. (B.5) can be written as:

\[
p_X(x) = \gamma_1^{q_1} \cdot (1 - \gamma_1). \tag{B.7}
\]

When \( N = 1 \) (i.e., only one task is in the product), value \( x = (q_1 + 1) \cdot w_1 \), which means \( x \) equals multiples of \( w_1 \). No matter how many times task \( v_1 \) fails, as soon as it succeeds, the one-task product gets manufactured. Term \( \gamma_1^{q_1} \cdot (1 - \gamma_1) \) in Eq. (B.7) is the joint probability of the \( q_1 \) times failures and the one time success.

When \( N \geq 2 \), value \( x \) is a linear combination of \( w_1 \) to \( w_N \) as shown in Eq. (B.3). For every combination \((q_1, \ldots, q_N) \in \Phi_{X=x}\), although they derive the same value \( x \), they represent different situations of failures. Therefore, a combination \((q_1, \ldots, q_N)\) represents a failure situation. For each failure situation, set \( \Delta \) contains all failure sequences under this situation. A failure sequence describes the sequence of failures, i.e., the sequence in which failures occur. In an example when \( N = 2 \), there are two sequential tasks \( v_1 \) and \( v_2 \) that must be executed to manufacture a product. One possible failure situation is \((q_1 = 2, q_2 = 1)\). It means that task \( v_1 \) fails twice and task \( v_2 \) fails once. Set \( \Delta \) under this situation contains three failure sequences: \( v_1 - v_1 - v_2, v_1 - v_2 - v_1, \) and \( v_2 - v_1 - v_1 \). Therefore, \(|\Delta| = 3\). They all result in the same value \( x \) by Eq. (B.3):

\[
x = (2 + 1 + 1) \cdot w_1 + (1 + 1) \cdot w_2
\]

even though they are different failure sequences.

Failures at the same task have no difference. Term \((\sum_{i=1}^{N} q_i)!\) in Eq. (B.6) is the number of failure sequences only if all failures are different. Term \((\sum_{i=1}^{N} q_i)!\prod_{i=1}^{N} q_i!\) is the actual number of failure sequences. It is derived by excluding indistinguishable failure sequences caused by multiple failures at the
same task from term \((\sum_{i=1}^{N} q_i)!\). In the \(N = 2\) example, for failure situation \((q_1 = 2, q_2 = 1)\), the number of distinct failure sequences is obtained by:

\[
|\Delta| = \frac{(q_1 + q_2)!}{q_1!q_2!} = \frac{(2 + 1)!}{2!1!} = 3.
\]

According to the rework policy, every failure at task \(v_i\), where \(1 \leq j < i \leq N\), means a success at task \(v_j\). Term \(\sum_{k=i+1}^{N} q_k\) is the number of successes at task \(v_i\) caused by failures at task \(v_k\), where \(i < k \leq N\). Note that, a product can only be manufactured when all tasks uninterruptedly succeed from the first to the last task. Therefore, task \(v_i\) must succeed at least once. Term \((\sum_{k=i+1}^{N} q_k + 1)\) in Eq. (B.6) is the total number of successes at task \(v_j\). Hence term \(\prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} q_k + 1}\) is the joint probability of successes at all tasks. Term \(\prod_{i=1}^{N} \gamma_i^{q_i}\) in Eq. (B.6) is the joint probability of failures at all tasks.

Due to the independence of failures, the probability of any failure sequence in the set \(\Delta\) under the same failure situation is the same. A failure sequence implies successes, therefore, term \(\prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} q_k + 1}\) is the probability of a failure sequence in set \(\Delta\) under failure situation \((q_1, \ldots, q_N)\).

In summary, when \(N \geq 2\), term \((\sum_{i=1}^{N} q_i)!/\prod_{i=1}^{N} q_i!\) is the number of failure sequences under failure situation \((q_1, \ldots, q_N)\); term \(\prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} q_k + 1}\) is the probability of a failure sequence under failure situation \((q_1, \ldots, q_N)\); and set \(\Phi_{X=x}\) contains all failure situations that result in \(X = x\). Therefore, the probability of \(X = x\) equals the sum of the probabilities of failure sequences under each failure situation in set \(\Phi_{X=x}\). It can be calculated by Eq. (B.6). Hence Eq. (B.5) is the PMF of random variable \(X\).
Appendix C  
Derivation of Eq. (3.4)

In this appendix, we present a derivation of the expectation of random variable $W$ that is described in Sect. 3.2.2.

C.1 Approximate the Distribution of $X$ by an Exponential Distribution

Discrete random variable $X$ is the execution time of a chain of sequential tasks. As discussed in Sect. B, Eq. (B.5) is the PMF of random variable $X$. Appendix A discussed that the expectation of $X$ can be computed by Eq. (3.3). If task execution time $w_i$ at each stage $i$ is small enough, then the intervals between possible values of $X$ are also small. Therefore, the execution time $X$ can be viewed as a continuous random variable. We observed that, the probability mass function of $X$ can be approximated by the probability density function of an exponentially distributed random variable. The rate of the exponential distribution equals $1/E[X]$. Figures C.1 and C.2 demonstrate such approximation.

C.2 The Expectation of the Maximum of Exponentials

Let random variables $W_1, \ldots, W_H$ be independent exponentially distributed random variables with the same rate $\lambda = 1/E[X]$, $Y = \max(W_1, \ldots, W_H)$. It is proved in [2] that the expectation of $Y$ is given by the follow equation:

$$E[Y] = \frac{1}{\lambda} \cdot \left( 1 + \frac{1}{2} + \ldots + \frac{1}{H} \right)$$  \hspace{1cm} (C.1)
Fig. C.1 The probability mass function derived from simulation results (left) and the probability density function approximated by an exponential distribution (right) for a 12-stage chain of sequential tasks with task execution times less than 1 time unit.

\[
E[Y] \approx \max_{j \in \{1, \ldots, H\}} \{E[W_j]\} \cdot \sum_{j=1}^{H} \frac{1}{j}.
\]  

(C.3)

Based on the discussed in Sect. C.1 and a worst case estimation, we approximate the distributions of execution times for parallel parts with an identical exponential distribution whose rate equals \(\lambda = \max(1/E[W_j])\), where \(j \in \{1, \ldots, H\}\). Hence we estimate the execution time of all the parts by Eq. (3.4), which is repeated here:

\[
E[Y] = E[X] \cdot \sum_{j=1}^{H} \frac{1}{j}. 
\]  

(C.2)

We can obtain \(E[Y]\) from \(Y\)’s PMF and CDF, which requires the PMF and CDF of each \(W_j\). Rigorous computation is too complicated to be realized in real time. Therefore, we adopt Eq. (3.4) for simplicity and fast computation.
Fig. C.2 The probability mass function derived from simulation results (left) and the probability density function approximated by an exponential distribution (right) for a 12-stage chain of sequential tasks with task execution times no less than 30 time units.
Appendix D
Introduction to SVR

Support-vector machines (SVMs) [3] have been extensively used for classification and regression [4, 5]. The key idea of an SVM is based on computing a linear regression function in a high-dimensional feature space by mapping the input data via a kernel function. It has been shown that an SVM offers major advantages in high-dimensionality space since its optimization is less dependent on the dimensionality of the input space. Compared to the empirical risk minimization (ERM) principle adopted in neural networks, the structural risk minimization (SRM) principle in SVM has greater generation ability [6]. It is robust in recognizing subtle patterns from complex data sets, even with corrupted data. Applications in pattern recognition [7, 8], vision-based head recognition [9] and vehicle detection [10] have already demonstrated that SVM is rigorous from a theoretical point of view and can lead to superior performance and great potential.

The application of SVM to regression is called support vector regression (SVR). The tutorial in [11] describes the process of selecting algorithms for training SVR models. We are therefore motivated by the many successful applications of SVR models (discussed in Sect. 4.1). We have found that SVR achieves the highest prediction accuracy for our testing data as compared with other algorithms such as linear regression and neural networks.

We first provide a brief introduction to support vector machine (SVM) in regression approximation [12, 13]. The basic idea of SVM is to map the training data from the input space into a high-dimensional feature space via a function \( \phi \) and then construct a separating hyperplane with maximum margin in the feature space. The set of training data points is denoted as \((x_1,y_1), (x_2,y_2), \ldots, (x_l,y_l)\), where \( x_i \in X \subseteq \mathbb{R}^n, y_i = \pm 1 \) denotes two class labels, and \( l \) is the number of training data. An SVM will find a hyperplane direction vector \( w \) and an offset scalar \( b \) such that \( f(x) = w \cdot \phi(x) + b \geq 0 \) for positive samples and \( f(x) = w \cdot \phi(x) + b \leq 0 \) for negative samples, which is illustrated in Fig. D.1. Therefore, even if a linear function cannot
be found in the input space to decide which class the given data belongs to, the function of $\phi$ can be used to find an optimal hyperplane that can clearly distinguish between instances of two classes.

SVR is derived from the idea of an SVM. The set of training data points is denoted as $(x_1,y_1), (x_2,y_2), \ldots, (x_l,y_l)$, where $x_i \in X \subseteq \mathbb{R}^n$, $y_i \in Y \subseteq \mathbb{R}$, $l$ is the number of training data. These data points are randomly and independently generated from the same distributions. An SVM approximates the decision function using the following form:

$$f(x) = w \cdot \phi(x) + b$$

where $\phi(x)$ represents the high-dimensional feature space that is obtained through a nonlinear mapping from the input space $x$. The hyperplane direction vector $w$ and offset scalar $b$ are estimated by minimizing the regularized risk function (D.2):

$$\min R_{\text{reg}}(f) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i))$$

where

$$L_\varepsilon(y_i, f(x_i)) = \begin{cases} |y_i - f(x_i)| - \varepsilon, & |y_i - f(x_i)| \geq \varepsilon \\ 0, & \text{otherwise.} \end{cases}$$

The first term $\|w\|^2$ of the regularized risk function $R_{\text{reg}}(f)$ is called the regularized term. Minimizing $\|w\|^2$ makes the risk function as flat as possible, thus playing the role of controlling the function regularization capacity. Function $L_\varepsilon(\cdot)$ is the loss function [14]. The second term $(1/l) \sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i))$ is the empirical error measured by the $\varepsilon$-insensitive loss function. The loss function provides the advantage of using sparse data points to represent the regularized risk function (D.2). Parameter $C$ is the regularization factor, which determines the penalty of a data point at the wrong side of the hyperplane, and $\varepsilon$ is called the tube size, which determines the data inside the $\varepsilon$ tube to be ignored in regression. They are user-prescribed parameters and need to be empirically optimized while building the model.
In order to get estimates of $w$ and $b$, the regularized risk function (D.2) is transformed to the primal objective function (D.4) by introducing the positive slack variables $\xi_i$ and $\xi_i^*$, which are used to measure errors outside of the $\varepsilon$ tube.

$$\minimize R_{reg}(f) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

subject to

$$y_i - w \cdot \phi(x_i) - b \leq \varepsilon + \xi_i$$
$$w \cdot \phi(x_i) + b - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, \cdots, l$$

$$\xi_i^{(*)} \geq 0$$  \hspace{1cm} (D.4)

where $\xi_i^{(*)}$ denotes variables with and without $\varepsilon$.

Finally, by introducing Lagrange multipliers and exploiting the optimality constraints, the decision function (D.1) has the following explicit form:

$$f(x) = \sum_{i=1}^{l} \left(a_i - a_i^*\right) K(x_i, x) + b$$  \hspace{1cm} (D.5)

In function (D.5), $a_i^{(*)}$ are Lagrange multipliers. They satisfy the equalities $a_i a_i^* = 0$, and $a_i^{(*)} \geq 0$ where $i = 1, \cdots, l$, and they are obtained by maximizing the dual of the primal objective function (D.4), which has the following form:

$$W(a_i^{(*)}) = \sum_{i=1}^{l} y_i \left(a_i - a_i^*\right) - \varepsilon \sum_{i=1}^{l} y_i \left(a_i + a_i^*\right)$$

$$-\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \left(a_i - a_i^*\right) \left(a_j - a_j^*\right) K(x_i, x_j)$$  \hspace{1cm} (D.6)

with the following constraints:

$$\sum_{i=1}^{l} \left(a_i - a_i^*\right) = 0, \quad 0 \leq a_i^{(*)} \leq C, \quad i = 1, \cdots, l.$$

Vector $w$ is written in terms of data points as:

$$w = \sum_{i=1}^{l} \left(a_i - a_i^*\right) \phi(x_i)$$  \hspace{1cm} (D.7)
**Table D.1** Common kernel functions [15]

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Function $K(x_i, x_j)$</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$x_i \cdot x_j$</td>
<td></td>
</tr>
<tr>
<td>Polynomial</td>
<td>$(x_i \cdot x_j + 1)^d$</td>
<td>$d$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\exp(-\frac{1}{\sigma^2}(x_i - x_j)^2)$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Radial Basis Function (RBF)</td>
<td>$\exp(-\gamma(x_i - x_j)^2)$</td>
<td>$\gamma$</td>
</tr>
</tbody>
</table>

**Fig. D.2** SVR to fit a tube with radius $\varepsilon$ to the data and positive slack variables $\xi$ measuring the data points lying outside of the tube.

Function $K(x_i, x_j)$ is defined as the kernel function [15]. The value of the kernel is equal to the inner product of two vectors $x_i$ and $x_j$ in the feature space $\phi(x_i)$ and $\phi(x_j)$, that is, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. The reason for applying kernel function is that SVR can perform a dot product in high-dimensional feature space using low-dimensional space data without knowing the transformation $\phi$. Any function that satisfies Mercer’s condition [16] can be used as the kernel function.

Some commonly used kernel functions are shown in Table D.1. We experimented with these kernels and the RBF kernel demonstrated the best performance.

After solving the value of $w$ in terms of the Lagrange multipliers, variable $b$ can be computed by applying the Karush-Kuhn-Tucker (KKT) conditions [17]. Based on the KKT conditions, only a few coefficients $(a_i - a_i^*)$ in decision function (D.5) have nonzero values, and the corresponding training data points have approximation errors equal to or larger than the tube with radius $\varepsilon$, and these data points are referred as support vectors. For all points inside the $\varepsilon$ tube, the Lagrange multipliers that equal to zero do not contribute to the decision function. Lagrange multipliers may be nonzero values and used as support vectors only if the requirement $\varepsilon_i \leq |f(x_i) - y_i| \leq \xi_i$ is satisfied (shown in Fig. D.2). In this case, the product of the Lagrange multipliers and constrains has to be equal to 0.

\[
\begin{align*}
    a_i (\varepsilon + \xi_i - y_i + (w, x_i) + b) &= 0 \\
    a_i^* (\varepsilon + \xi_i^* + y_i - (w, x_i) - b) &= 0 \\
    (C - a_i^{(*)})\xi_i^{(*)} &= 0
\end{align*}
\]  

(D.8)
Since $a_i^{(*)} = 0$, and $\xi_i^{(*)} = 0$ for $a_i^{(*)} \in (0, C)$, $b$ can be computed as:

\[ b = y_i - (w, x_i) - \xi_i \quad \text{for} \quad a_i \in (0, C) \]
\[ b = y_i - (w, x_i) + \xi_i \quad \text{for} \quad a_i^* \in (0, C). \] (D.9)

Therefore, as seen from the decision function (D.5), only the support vectors are used to determine the function value, as the values of $(a_i - a_i^*)$ for the other training data points are equal to zero. This highlights the main advantage of SVR, i.e., support vectors are usually only a small subset of the training data points; this is referred to as the sparsity of the solution.

In summary, in order to train a SVR model, we need to optimize the kernel function $K(\cdot)$, the regularization factor $C$, and the tube size $\varepsilon$ through the procedures described in tutorial such as [11].

References


