A Design Method of Performance Test Case Based on kNN Algorithm

Ji-jun ZENG¹, Fa-mao MEI² and De-xin HUANG³

¹Information Center Guangdong Power Grid Co. Ltd, Guangzhou 510600, China
²CSG-key Laboratory of Information Technology Testing, Guangzhou 510600, China
³CASIC-CQC Software Testing and Evaluation Technology (Beijing) Corporation, Ltd., Guangzhou 510600, China

Keywords: kNN Algorithm, Performance Test, Test Case.

Abstract. This paper based on existing historical test case set to combined with the kNN algorithm to train the classification model. The model is used to assist the design of the software performance test to obtain the optimal performance test case set. This method can provide reference for software performance test case design.

Introduction

The massive accumulation of data and the increase in system computing power make the infiltration of artificial intelligence into various industries, especially software testing. Although the current testing tools are partially automated, still need to manually set some parameters and configuration items. In particular, performance test case design, such as execution path, number of concurrent, duration, etc., still rely on the experience of testers to manually process. But AI allows testers to reduce a lot of repetitive work and make test work smarter, more efficient and more comprehensive.

Current Situation

Microsoft's Microsoft Neural Network algorithm is used for defect classification or regression testing tasks, and can predict output based on certain input properties. King.com uses Monte Carlo tree search algorithm, automatic heuristic construction algorithm, and enhanced topology neuron evolution algorithm for functional testing and stability testing of games.

However, the principles of these commercial test tools, especially those related to performance testing, are too complex or not yet public. This paper intends to explore the application of kNN algorithm principle in performance testing from the perspective of performance testing, and summarizes the performance test case design method.

Related Conception

kNN (k-NearestNeighbor) is a supervised learning algorithm that obtains the categories of test samples by inputting test data and training sample data sets. When the training data set and the three elements are determined, it is equivalent to dividing the feature space into some subspaces. For each training instance, all points closer to the point than the other points form an area, and the category of each area is composed of Decision rules are determined and unique to divide the entire area. For any one test point, find the subspace to which it belongs, and its category is the category of the subspace.

The distance measure in this paper uses the Minkowski Distance, defined as: \( D(x,y)=(\sum_{i=1}^{m}|x_i - y_i|^p)^{1/p} \), like \( p\geq 1 \), when \( p=2 \), is euclidean metric, when \( p=1 \), is Manhattan Distance.
The choice of k in the algorithm can have a significant impact on the outcome of the algorithm. When the value of k is small, over-fitting may occur; when the value of k is large, under-fitting may occur. So in the application, generally choose a smaller odd number.

1. Set the parameter k; maintain a priority queue of size k from the largest to the smallest for storing the nearest neighbor training tuple. Randomly select k tuples from the training tuple as the initial nearest neighbor tuple, calculate the distance from the test tuple to the k tuples, and store the training tuple labels and distances into the priority queue.

2. Traverse the training tuple set, calculate the distance between the current training tuple and the test tuple, and obtain the distance L from the maximum distance Lmax in the priority queue; if \( L \geq L_{\text{max}} \), discard the tuple and traverse the next element. If \( L < L_{\text{max}} \), the tuple of the maximum distance in the priority queue is deleted, and the current training tuple is stored in the priority queue.

3. After the traversal is completed, calculate the majority of the k tuples in the priority queue and use them as the category of the test tuple.

4. Calculate the error rate after the test tuple set is completed, continue to set different k values and re-train, and finally take the k value with the smallest error rate.

**Classify Regulation**

The classification decision rule in kNN is usually a majority vote, that is, the class of test samples is determined by the majority of the k adjacent samples of the test sample.

The majority voting rule is as follows: Given test sample \( x \), the nearest k training instances form a set \( N_k(x) \), classification loss function is 0-1 loss. If covered area \( N_k(x) \) is \( c_j \), or the classification loss is

\[
\frac{1}{k} \sum_{x_i \in N_k(x)} I\{y_i \neq c_j\} = 1 - \frac{1}{k} \sum_{x_i \in N_k(x)} I\{y_i = c_j\}.
\]

To make the classification error rate small, that is, the empirical risk is the smallest, the majority vote is equivalent to the empirical risk minimization. But kNN Mode is equal to arbitrary x to get \( N_k(x) \), loss function is 0-1, optimization strategy minimizes empirical risk. In general, use majority vote on \( N_k(x) \).

**Classify Mode**

Enter test data \( x \) and training data set \( T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \), and \( x_i \in \mathbb{R}^n, y_i \in \{c_1, c_2, ..., c_k\} \).

The output instance \( x \) belongs:

First, based on the given distance metric, find the k samples closest to the \( x \) distance in the training set \( T \). Cover the k point to x’s as \( N_k(x) \)
Then in \( N_k(x) \), according to Classify Regulation to make sure x category y, classify mode as follows:

\[
y = \arg \max_j \sum_{x_i \in N_k(x)} I\{y_i = c_j\}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, k
\]

Find the required categories and their characteristic attributes from the data set based on the model.

**Application**

In this paper, we use the classification function of kNN algorithm to test whether the test case can test the system performance bottleneck and improve coverage as the label; calculate the optimal distance metric and k value in the model; analyze the optimal adjacent test case feature attributes, auxiliary use case The generation.

**Historical Test Case Selection and Analysis**

From the existing test cases, select the test cases that are better and can test the system performance bottleneck. Make up training testing case set \( T \). \( T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \), \( x_i \) is cases, \( y_i \) is classify labels. The following table is a brief performance test case.

<table>
<thead>
<tr>
<th>Test case ID</th>
<th>XXX-XX</th>
<th>Test case name</th>
<th>XXXX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test case description</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test case ID</th>
<th>Test scenes</th>
<th>Concurrent number</th>
<th>Duration</th>
<th>...</th>
<th>Testing steps</th>
<th>Expected results</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>10h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Analyze the cases and choose concurrent numbers, duration and testing steps in many attributes as \( x_i \)’s feature attribute set \( F = \{\text{concurrent numbers, duration, testing steps} \ldots\} \).

**Use Case Classification and Feature Attribute Value Extraction**

Calculate the distance metric \( D(x, y) \) according to the distance metric principle and the attribute set \( F \) above; then train the model to find the best k value by inputting the training data; secondly, calculate the test sample and all the training samples by the optimal model The distance, according to the category of the recent k training samples, the majority of the voting methods are used to classify the test cases that can test the performance bottleneck of the system and improve the coverage; At last, extract these test cases’s feature attribute set \( FV = \{v_1, v_2, \ldots, v_n\} \).

**Optimize Test Cases**

In the design of the new performance test case, the set of feature attribute values \( FV \) is used as a reference to the value of the feature attribute set \( F \) to assist in the design of the test case.
Effect Example

Table 2. Effect example table.

<table>
<thead>
<tr>
<th>Concurrent Number</th>
<th>Duration</th>
<th>The total number of HTTP requests made</th>
<th>Server rejects</th>
<th>Return 500 error counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>1h</td>
<td>10300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>1h</td>
<td>20600</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>10h</td>
<td>20600</td>
<td>10</td>
<td>201</td>
</tr>
<tr>
<td>1200</td>
<td>10h</td>
<td>40880</td>
<td>1412</td>
<td>839</td>
</tr>
</tbody>
</table>

Epilogue
Through the analysis of KNN algorithm and introduce it into the design of performance test cases, based on the small sample test data, the key feature attribute sets are obtained; and in the process of system performance test, these feature sets are used to design The optimal test case reduces the workload of the tester and saves costs.

References