Mouse Wait Classification

Yingyin A Xiao
yix193@ucsd

Sijie Mei
simei@ucsd.edu

Pan Yeung
payeung@ucsd.edu

Abstract

This thesis describes an study of machine learning and its application to mouse wait time in computers. Specifically, we are building a classification model of mouse wait time based on dynamic and static system information within the 2020 time interval to classify if a mouse wait event would last within 0-5 secs, 5-10 secs, or 10+ secs. Dynamic system information, such as CPU utilization, is subject to the configuration of each system. Therefore, by incorporating static system information which includes the computer configuration of each system into the model, we could significantly improve the accuracy of the prediction. Currently, the model reaches an accuracy of 70% with Decision Tree Classifier.

1. Introduction

Mouse wait is officially called Windows wait cursor, also known as busy cursor, which is a mouse icon status that indicates that mouse cursor is busy or processes are busy performing operations and set the cursor to the busy state [1]. When the mouse icon changes to Windows wait cursor status, spinning wheel incident happens and causes users to wait for current mouse wait to finish before interacting with applications. The problem is that most mouse wait events are unexpected and users are not willing to wait if it lasts a longer than 1-2 seconds. If we are able to predict the mouse wait time, users could terminate their processes ahead of time when they know it might be a long-term wait. Currently, most studies online is about fix or prevent mouse wait event, but there is no research conducted on the mouse wait time prediction, which tells users the expected mouse wait times.

In our study, we will build a classification model to predict mouse wait by separating the wait time into three groups: 0-5 sec, 5-10 sec, and 10+ sec. The reason we classify the time in this way is that we regard 0-5 sec as short wait time, 5-10 sec as medium wait time, and 10+ sec as long wait. Features of our model include dynamic system information and static system information, which will be introduced in the following sections. So far, as for the model, Decision Tree Classifier has the highest accuracy with fair model evaluation.

2. Data Collection

Intel® System Usage Report (SUR) collector XLSDK is a framework that used to collect data for this project. In the data collection process, we implement input libraries (IL) that collect comprehensive raw data from computer and then implement Analyzer Task Libraries (ATL) to clean and process these raw data.

At the beginning, Intel® Energy Checker SDK Energy Server (ESRV) executes IL files to collect samples every second. After ESRV finishes collecting data, it writes raw data into Database (DB) file. Next, when ATL file is executed, it reads the DB file and processes raw data collected from the ILs, and outputs data into log files corresponding to each GUID (Global Unique Identifier for each computer). In the end, log files are uploaded to online server for further analysis.

User_wait input library is a Dynamic Link Library (DLL) file. During the collection process, ESRV executes this file's function every 0.1 second and also at any moment when user click any mouse button. At each iteration, User_wait IL uses Windows API GetCursorInfo and GetIconInfo to capture and analyze the mouse icon. GetCursorInfo returns the position of cursor and a handler to cursor. By inputing this handler to GetIconInfo, it will return the icon status of cursor. When it returns a Windows wait cursor, ESRV records the current time and mouse wait event. After ESRV finishes its collecting process, it outputs raw mouse wait data to DB file with variable name and variable type defined in User_wait IL. Since samples are collecting at every 0.1 second. Any event less than 0.1 second can't be detected. However, these events are transient and human can't even perceive them, so missing these

events will be of no significance.

After recording the mouse wait event, we also need to know the dynamic system information when events are happening. It will be the best if we can collect system usage of the process that is responsible. However, it is hard to tract which application causes the mouse wait event since OS can cause a mouse wait at any time. Therefore, we decide to collect all processes' system usage when mouse wait event happens. Besides User_wait IL, we have Process IL. ESRV executes the Process IL to record comprehensive information of each process. At each iteration, It calls the Windows API ZwQuerySystemInformation and OpenProcess. ZwQuerySystemInformation will list all processes and its PID (process ID) currently running on the system and OpenProcess utilizes PID to collect each process's system usage like process name, running time, I/O usage, memory usage, disk usage and page fault. Unlike User_wait IL which records data only when it encounters spinning wheel incident event, the Process IL records and writes all data it captures every second.

After the raw data is collected and saved as raw DB file, analyzer tasks read the file and analyze them by language C and SQL. Since all measurements by ILs are timestamped, the start time (ts) of a mouse wait event is the timestamp of the first row of that mouse wait event. The duration (wait_msec) of the mouse wait event is the timestamp difference between the first row and last row of that mouse wait event. By using the ts of mouse wait event, ATL can locate the processes in plist data set that has the same ts, which are running processes during mouse wait event, and compute CPU-util, Disk-Util, Network-util and Hard Page Fault. These features will be saved as mouse_wait_all log file. In the file, each row denotes a mouse wait event and each column denotes system usage related to one mouse wait event.

	number of rows	number of Guid
Whole (2020)	14,534,433	29,587
Train (2020.Oct)	1,729,282	16,778
Test (2020.Nov)	1,488,682	15,702

Table 1. Data set

At the beginning, we implement our own XLSDK and ATLASK and do the data collection on local computers, but then we realize the scale of data sets is too small and may cause the model has high bias. We need more data from all kinds of users. Since we don't have resource to access such many users, Intel team collect and provide us with data sets. Data sets includes 14,534,433 rows with 29,587 unique GUID within the 2020 interval. Since the complete dataset is fairly large, we choose all rows within 2020 October interval as train set and choose all rows within next month, 2020 November, as test set for our model. In the train set, there are 1,729,282 rows and 16,778 distinct GUID. In the test set, there are 1,488,682 rows and 15,702 distinct Guid. For the target feature wait_msecs, we divide the wait time into 0-5s, 5-10s and 10+s as a preparation for the classification model. After exploring the correlation between potential features and the mouse wait time, we find that dynamic features, including CPU utilization, disk utilization, hard page faults, and static features, including the number of cores, RAM, model type, etc, could influence the mouse wait time. These features are then incorporated in the model.

3. Exploratory Data Analysis

In general, mouse wait is caused by cursor that indicates that an application is busy performing an operation [1]. This means system usage is overused. Before the prediction task, we want to firstly explore what factors induce mouse wait. We will do some exploratory data analysis on both dynamic system info and static dynamic info.

3.1. Dynamic System Info

The data set we use is "mousewait_all.csv001", which is provided by the Intel teams. This data set records system usage information before and after mouse wait happens. Each feature in this data set consists of prefix, infix and suffix. Prefix has "before" and "after". It represent is this feature recorded before or after mouse wait event. Infix has "CPU_Util", "harddpf", "disk_util" and "network_util". This represents what kind of system usage this feature records. Suffix has "min", "max" and "mean". This represents the way this feature computes statistics.

Before_CpuUtil_Max
Before_HardDPF_Max
Before_DiskUtil_Max
Before_NetworkUtil_Max

Table 2. 4 Dynamic System Info Features We Chose

We chose 4 Before_***_Max features. The reason we choose "Before" is that our model is a predictive model, which need to finish prediction before the predictive target happens, so it is impossible to get "after" data. The reason we chose "Max" is that mouse wait is caused by system usage overused, "Max" best fits this situation.

The first step of the exploratory data analysis is data cleaning. There is no Null value in the dataset. However, the column "wait_msecs" contains a lot of

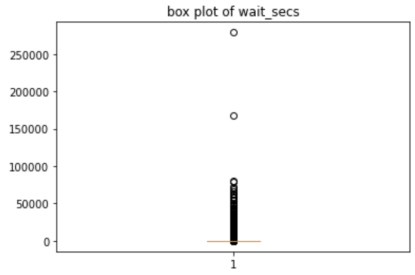

Figure 1. Outlier Exists

outliers and its distribution is not normally distributed. As seen in Figure 1, we see there is several points overwhelmingly larger than others that are accumulated at less than 3000 sec. However, we decide not to remove these outliers because the long wait is crucial in our study. Considering "before_cpuutil_max" (maximum cpu utilization before the mouse wait state) is collected in percentage, we remove rows that are larger than 100.

After data cleaning, we work on data transformation. Features like "before_harddpf_max" (maximum hard page faults before the mouse wait state) have large variance. It has an exponential growth and make its data pattern of the first half not distinguishable. To solve this problem, we execute a log transformation on "before_harddpf_max", "before_cpuutil_max", "before_diskutil_max", "and before_networkutil_max".

$Data_{New} = \log_2 Data_{Old}$

To find potential features for the mouse wait time, we draw scatter plots in Figure 2 to see if the log-transformed "before_harddpf_max", "before_cpuutil_max", "before_diskutil_max", and "before_networkutil_max" have any relationship with wait_secs. CPU utilization before the mouse wait state seems to have a slight linear correlation with mouse wait time. As "before_cpuutil_max" gets larger, the wait time gets larger. In the second graph "wait_msec VS hard page fault" and third graph "wait_msec VS disk utilization", we can see it exists lines that separate long time mouse wait and short time mouse wait. Long time mouse wait events cluster at hard page fault's interval 5 - 12 and disk utilization's interval 10 - 20. These separable lines can be used in decision tree to classify short time mouse wait event and long mouse wait event, so we should include these features into our model. Although it seems that there is also separable long time mouse wait and short time mouse wait in fourth graph "wait_msec VS network ultilization", when we check the distribution of network utilization, 99.89% of network utilization are 0, which means this feature almost has just one value and so wouldn't be helpful for building model. Based These scatter plots, we

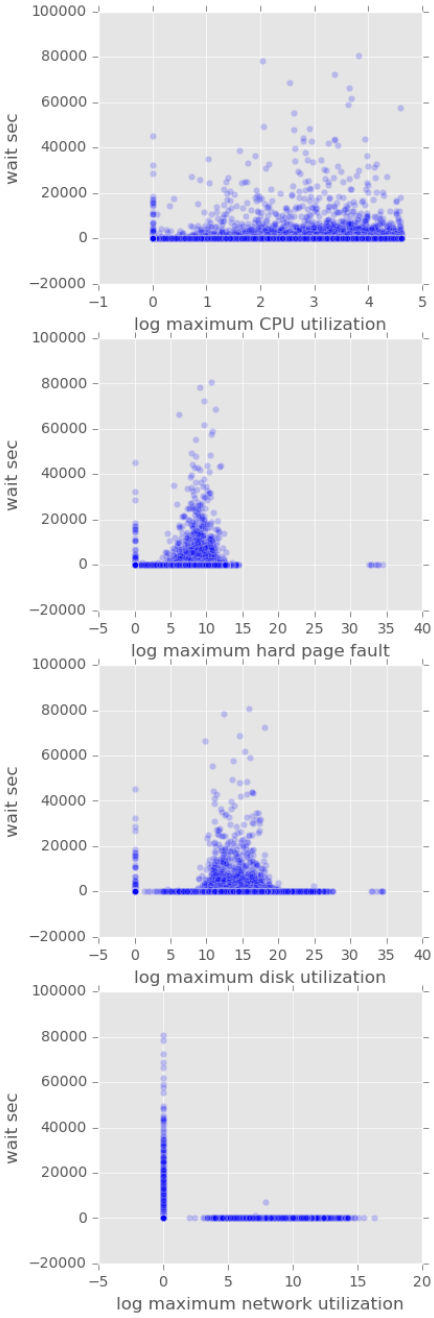

Figure 2. Scatterplot for CPU util, disk util, and harddpf

choose "before_harddpf_max", "before_cpuutil_max", "before_diskutil_max" as our dynamic system info input features.

We then create a new column for the classification task, 0 - 5 sec, 5 - 10 sec, and 10+ sec based on mouse wait time. The distribution is plotted at Figure 3. Generally, people regard waiting for the mouse less

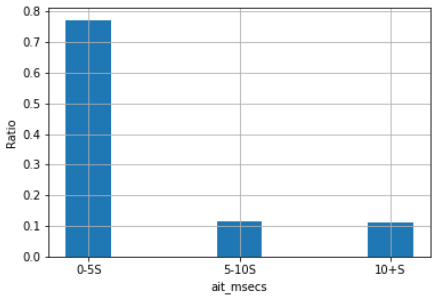

Figure 3. Wait_secs distribution

than 5 seconds as normal, 5 - 10 seconds as a little long, and over 10 seconds as a long wait. Figure 3 is the distribution of the wait time in October, 2020. 77% of the mouse wait events happened less than 5 seconds. That would cause a large bias on the first classification group, which is a problem that we would solve later.

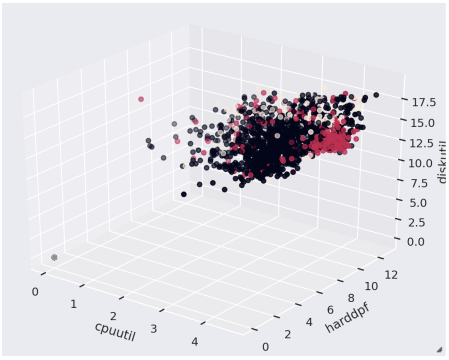

Figure 4. Wait_msecs VS Dynamic System Info Features

To see how "before_harddpf_max", "before_cpuutil_max", "before_diskutil_max", and "before_networkutil_max" might affect mouse wait time, we draw a plot in 3 dimensions in Figure 4 to explore further. Each axis represent one dynamic system info feature. Color of point represents its classification group. Black points represent 0 - 5 sec, red points 5 - 10 sec, and yellow points 10+ sec. Also, the 3D plot is created from 12 GUIDs in order to make the shape more obvious. In observation of the plot, first of all, the data in the three groups tend to be on a similar area, but the red points cluster when "before_harddpf_max" is at 6 - 10, "before_diskutil_max" at 10 - 12.5, and "before_cpuutil_max" at 3 - 5 (these features are log-tranformed). We can barely see yellow points on the plot because there is relatively few mouse wait event over 10+ sec. After rotating the 3D map, we can see from a certain perspective, classification groups are separated, which means there might be separable hyperplanes in certain projection. This reminds us machine learning models that utilize separable hyperplanes, like decision tree or SVM, will be good base model for predicting mouse wait.

3.2. Static System Info

The data set we use is "system sysinfo unique normalized.csv000", which is provided by the Intel teams. It contain 32 different features and 100,000 unique systems. This data set provides information about the system hardware like CPU model, GPU, ram, etc. We choose 9 features from the provided information, as shown below, as they rigorously represent the configuration of each system. Among the 9 features, "chassistype", "os", "graphicscardclass", "cpucode" and "persona" are nominal features, and "ram", "age_category", "processornumber" are quantitative features as they are ordinal.

chassistype
ram
os
ofcores
age_category
graphicscardclass
processornumber
cpucode
persona

Table 3. 9 Static System Info Features We Chose

Then, we use Chi-squared test to test if each of the nominal features are correlated with the mouse wait time. The Null hypothesis is that the observed frequencies of each categorical variables match the expected frequencies. The Chi-squared test is conducted with bootstrapping. We sample 5000 mouse wait events from the training set and compute the p-value for each categorical feature. We then repeat the process 1000 times. With 1000 p-values in hand, we calculate their mean and median for each feature. In Figure 5, it turns out that only the statistics of CPU code, os and persona are less than the significance level 0.05, the null hypothesis are rejected, meaning that the features are not independent. That gives us an idea that the two categorical features do have a relationship with mouse wait time.

The features are incorporated into the model later as an improvement. However, 'cpucode', 'os' and 'persona' are are nominal and can't be directly input in model. For these nominal features, we execute a one hot transformation. One hot transformation creates a list of a list of length N, the amount of distinct values for each

	mean	median
chassistype	0.244568	0.150854
os	0.174335	0.058439
graphicscardclass	0.400003	0.355233
cpucode	0.000540	0.000000
persona	0.047606	0.004217

Figure 5. Chi-squared test on each categorical feature

column. If this column has the Kth distinct value, the Kth value in list will be 1 and value in other positions are 0.

After One Hot transformation, the 3 nominal features are transformed into 907 quantitative features in a matrix. However, the size of the 907 features is too large to do calculations and most of the values inside the matrix are zeroes as sparse matrix. Therefore, we executed a PCA(Principle Component Analysis) transformation with n = 30. This PCA will pick 30 most meaningful dimensions and project other columns on these 30 dimensions. These 30 columns will be our static system info input features.

4. Classification Model

Target feature: wait_msecs.

Target feature classification groups: [0-5 sec, 5-10 sec, 10+ sec].

4.1. Baseline

Before building the baseline model, we conduct some feature engineering. Since there are initially some zeroes in the features "before_harddpf_max", "before_cpuutil_max", "before_diskutil_max", after they are log-transformed, the zeroes become negative infinities. We decide to replace the negative infinities with zeroes. It could cause little bias to the model because such values only occupy around 2% of the whole dataset.

We build the baseline model based only on the dynamics system information using Decision Tree Classifier with default parameters. We select data that in the October, 2020, where it has 2,034,448 mouse wait events with 18,058 unique GUIDs, and hope to predict the mouse wait on November, 2020. Thus, we use data in October, 2020 as the training set and data in November, 2020 as the test set. The test set includes 2,034,448 mouse wait events and 18,058 unique GUIDs. Below is the result of the model in both the training and test set.

	precision	recall	f1-score
0-5S	0.985	1.000	0.992
10+S	0.996	0.955	0.975
5-10S	0.999	0.940	0.969

Figure 6. Accuracy of Baseline Model on Training Set

	precision	recall	f1-score
0-5S	0.773	0.756	0.764
10+S	0.119	0.129	0.124
5-10S	0.121	0.128	0.125

Figure 7. Accuracy of Baseline Model on Test Set

From the result, first of all, the model is overfitted, as the accuracy of the training set is exceptionally high, while that of the test set is relatively low. The problem could be mitigated by adjusting the depth of the decision tree. Also, the precision and recall are higher for the group 0-5s but lower for the rest of the two groups. That means the model performs better on the first group. The reason could be that there are much more mouse wait events that last 0 - 5 seconds. This problem could be improved by adding in more features, such as static features that take system configuration into account.

4.2. Improved Model

To further improve the model, we include features from static system information. We perform One Hot Encoding for the categorical features, append to the quantitative features and convert the data frame into a matrix. Lastly, combing the the matrix with the one from the dynamic features, we use Decision Tree Classifier to perform the modeling again on the same training and test set.

	precision	recall	f1-score
0-5S	0.792	0.774	0.783
10+S	0.173	0.189	0.181
5-10S	0.159	0.168	0.163

Figure 8. Accuracy of Model Combined Static System Info

The performance of the new model is obviously better than baseline model, but it still has the problem that F-1 score on group 5-10 second and group 10+ second is pretty low. Then, we improve the model by tuning the model's hyper parameters "Max_Depth" and "Max_Leaf_Nodes".

By testing on max_depth from 5 to 100, we find that as the maximum depth of the tree gets larger, the F-1 score of group 0-5 second gets lower, F-1 score of group 5-10 second and group 10+ second gets higher. We can choose a max_depth like 50 if we want to focus on group 5-10 second and group 10+ second. By testing max_leaf_node from 2 to 5200, we find as the maximum leave node of the tree gets larger, the F-1 score of group 0-5 second keeps the same F-1 score of group 5-10 second and group 10+ second gets higher. All 3 groups' F-1 score keep relatively constant after max leave node is after 4000. We can choose We can choose a max leaf node like 4500 to achieve a high F-1 score.

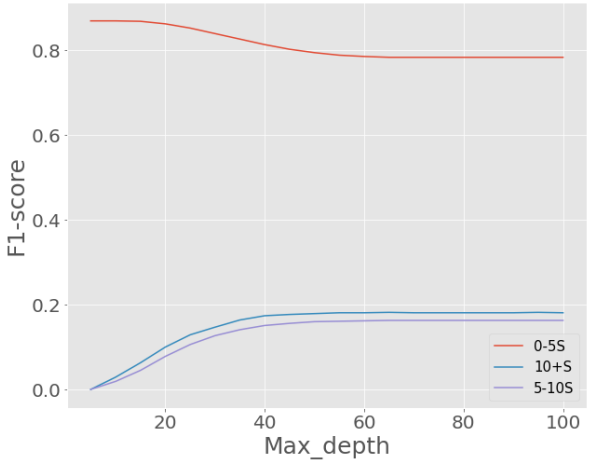

Figure 9. F1-score VS different Max_Depth

4.3. Bagging

Even by adding more features or changing different max_depth, the F-1 score on group 5-10 second or group 10+ second still can't exceed 0.2. We think this is because these two group has too little samples that each of their proportions is about 10% of whole data set. One possible way to solve the problem of unbalance distribution is to use bagging.

Bagging, also called bootstrap aggregating, is a machine learning algorithm designed to improve the stability of the machine learning model. Bagging starts by bootstrap samples from the training set and learns them in parallel. Doing so will generate multiple models, and we will pick the prediction that is the most frequent among the model outputs.

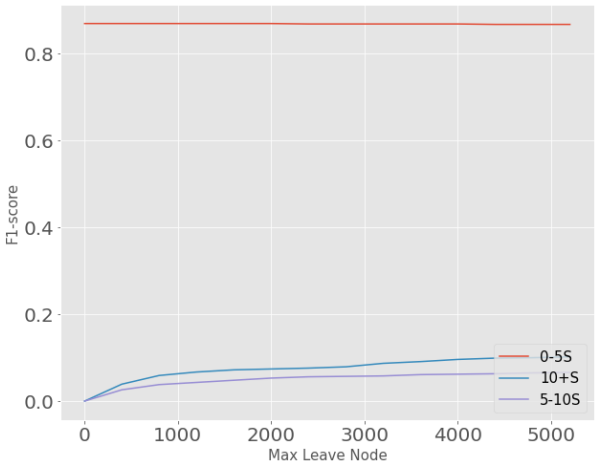

Figure 10. F1-score VS different Max_Leaf$_{N}ode$

In the figure 3, we see group 0-5 second is around 7 times big as group 5-10 second or group 10+ second. So we divide the 0-5 second group into 7 sub-group and combine each sub-group with two small groups. Then we have 7 new data sets. Each of new data set has the same sample amount, which ensures that the classification groups in new data set have close proportion. Then, We train 7 decision tree models with each new data set. After classification, the 7 models will give us 7 predictions and we use majority rules to decide the final prediction.

	precision	recall	f1-score
0-5S	0.825	0.622	0.709
10+S	0.183	0.334	0.236
5-10S	0.170	0.310	0.220

Figure 11. F1-score After Bagging

Bagging brings us a big improvement two on group 5-10 second and group 10+ second although the F-1 score on the group 0-5 second decreases. The point is that we want to pay more attention on these two groups with less amount of mouse wait events. To improve performance on all groups, we need to explore more related features.

5. Conclusion and Future Work

In this paper, we have explored how machine learning can be used for classifying the mouse wait time. The mouse wait is separated into three groups for classification: 0-5 seconds, 5-10 seconds, and 10+ seconds. We started by collecting data using Intel's software development kits, and by training a Decision

Tree Model on dynamic features, we achieved an accuracy of 61.2%. The model was improved to 73.9% by adding in static features which include information of system configuration. Unfortunately, due to highly unbalanced wait time as there is over 80% of data at less than 5 seconds, the second (5-10 seconds) and third groups (10+ seconds) were less accurate than the first group (0-5 seconds). This problem was mitigated by using parameter tuning and bagging algorithm, which overall increased the F1 score 2 times higher. The work is presented on a Github web page.

As future work, firstly, since there were few common GUIDs as we merged different data sets, we aim to collect data that involves the same GUIDs. Also, we will keep improving accuracy on the median and long mouse wait group by finding more relevant features.

References

[1] Wikipedia contributors, "Windows wait cursor — Wikipedia, the free encyclopedia," 2021. [Online; accessed 7-February-2021].