EP2420 Project 2 - Dimensionality Reduction

Cheng Yang Jie Huang

Dec, 2020

Project Overview

In networked systems engineering, the data collected from sensors and logs are usually high-dimensional which costs a great amount of time for the prediction. In order to reduce the computational overhead and maintain the same accuracy of prediction, we investigate and compare different methods with the objective to reduce the dimensionality of the features. Based on this, we select the top 2 features and map them in a 2-dimensional space to visualize the evolution of the system state. At last, we consider online feature selection and try to reduce the complexity of training in terms of time and features.

The project is performed using Python and the related packages are Numpy, SciPy, Matplotlib, Scikit-learn, and Pandas.

Background

We will briefly introduce the tree-based feature selection method used in this project, which is based on the random forest regression model and the features are ranked according to the feature importance.

Random Forest is a set of decision trees built on random samples with different policies for splitting a node [1]. Random forest uses the bootstrap method in building decision trees and there are two ways to interpret these results. The more common approach is based on a majority vote in the classification case and for the regression model, the average of each tree's prediction is used.

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. We can use the Scikit-learn library to easily build the Random Forest model and calculate the importance of different features.

The Random Forest Regression model is not only used in feature selection, but also in the prediction task. To evaluate the model accuracy of the predict model, we use the Normalized Mean Absolute Error (NMAE) as the metrics, which is defined as $NMAE = \frac{1}{\overline{y}}(\frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y_i}|)$, whereby $\hat{y_i}$ is the model estimation for the measured service metric y_i, and $\overline{y}$ is the average of the observations y_i.

Data Sets and Data Pre-processing

Data Sets Description

The traces used in this project is gathered from a testbed at KTH that runs a key-value(KV) store service under dynamic load. The structure of the system is shown in Figure 1.
Using machine learning techniques, we measure infrastructural statistics metrics X in the infrastructure and predict service-level end-to-end metrics Y. The feature X consists of $X_{cluster}$ and X_{port}, respectively coming from the core of the Linux operating system that runs the KV service application and the OpenFlow switches in the network[2]. The Y_{kv} metrics consists of two following statistics:
1) Read Response Time as the average read latency for obtaining responses of operations performed per

second;
2) Write Response Time as the average write latency for obtaining responses of operations performed per second.
The Read Response Time, marked as "$ReadsAvg$" in the data set, is used as the predicted variable in this project.

The data traces were generated according to two following load patterns. The detailed setup of the two patterns can be found in [3].
1) Periodic-load pattern: the requests was produced following a Poisson process whose arrival rate is modulated by a sinusoidal function over a period of 28962 seconds.
2) Flash-crowd load pattern: the requests were produced following a Poisson process whose arrival rate is modulated by the flash-crowd model described in [4].
The parameters of the data sets are given in Table 1.

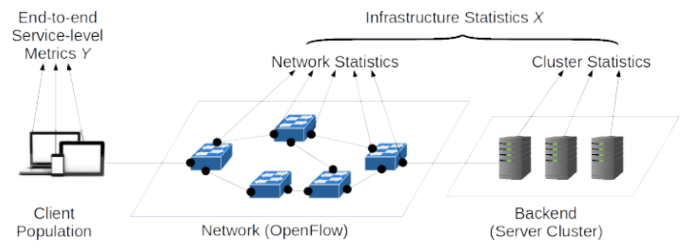


Figure 1: The testbed at KTH, providing the infrastructure for experiments where end-to-end service-level metrics were predicted from low-level infrastructure measurements[3].

Load Patterns	Number of features	Number of measurements
Periodic	1752	28962
Flashcrowd	1724	19444

Table 1: Features and measurements used in different load patterns

Data Pre-processing

For Tasks I, II, and III and the first part of Task IV, to reduce the computing time, only half of the traces are used. And then it is split into training and test samples with the proportion of 7: 3.

As for outlier removal, the main goal of the project is to select appropriate features from the data set. We will remove features with all zero values measurements in the training set or any data subset that we need to study. This is especially important for the ARR algorithm which needs to calculate the cosine similarity function with the vector norm as the denominator.

Task I - Comparative evaluation of different dimensionality reduction methods

The objective of this task is to compare different dimensionality reduction methods with respect to the accuracy and overhead of a predictor that learns on the reduced feature space.

1.1 Prediction accuracy study of four methods

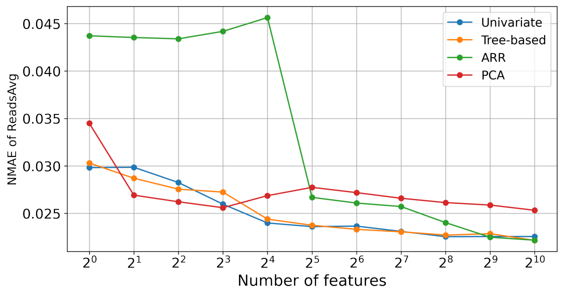
Firstly we use the KV period data set and set the metrics "$ReadsAvg$" as the target variable. Three methods, namely, univariate feature selection (which uses Pearson correlation), tree-based feature selection, and ARR(see in [5]) are used to select features from the training set. And then Principle Component Analysis (PCA) is used to generate principle components from the original training set. The first two methods are supervised learning approaches that make use of the target variable, while the ARR and PCA methods are unsupervised approaches.

For each feature selection method or PCA, we train a random forest regression predictor that learns using only the top k features or principle components. The prediction accuracy in function of k is shown in Figure 2(a). We can observe that there are two error trending modes. The first is the red line corresponding to PCA. The error first drops and then rises. We can take the turning point as k^*, which is 8. The second mode is shown in the lines corresponding to the remaining three methods. The error generally decreases as the number of features increases, but the decrease becomes insignificant after a certain point which can be set as k^*. For univariate, Tree-based, and ARR method, we set k^* as 256, 256, and 512 separately.The prediction results for "*ReadsAvg*" of the KV flashcrowd data set are similar, as shown in Figure 2(b). We can choose the proper k^* for KV flashcrowd in similar way and the result is shown in Table 2.

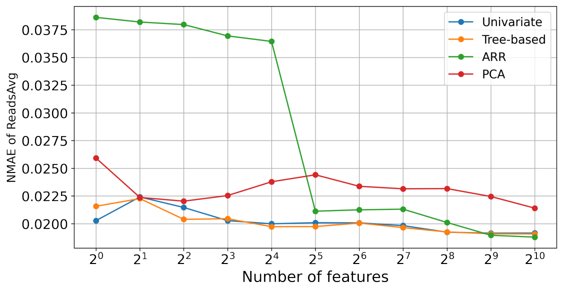
Load Pattern \Method	**Univariate**	**Tree-Based**	**ARR**	**PCA**
Periodic	256	256	512	8
Flashcrowd	256	256	512	4

Table 2: k^* selection for different load patterns of KV periodic and KV flashcrowd

Theoretically, for the three methods of directly selecting original features, when there are enough features (when k is larger than 512 in Figure 2), the error levels of those three methods tend to be the same, while PCA is different because it generates new component which replaces the original features. But we hope to use a smaller number of features to achieve a lower error level. If we want to use as few features as possible, we can choose tree-based and univariate method, and use at least 16 features to achieve a relatively low error level. If we want a very low error level, then more than 256 features are needed for tree-based and univariate method and 512 features for ARR. Under these two considerations, ARR is not a good method. But the advantage of ARR is that it is unsupervised and its interpretability is greater than PCA, because we can see which features can represent the input features through ARR directly.



(a) "*ReadsAvg*" estimation in KV periodic



(b) "*ReadsAvg*" estimation in KV flashcrowd

Figure 2: Random forest regression NMAE with increasing number of features or principle components

1.2 Feature overlapping of k^* feature

For each feature selection method, we identify a value k^*(as small as possible) so that the predictor achieves a low error on the data trace. Let's call the top k^* features the k^* feature set of the the particular feature selection method for a specific trace. We compare the k^* feature sets of the three methods regarding overlap by constructing a heat map. Each cell in the map corresponds to the number of joint features for two (trace, method) pairs.

The result is shown in Figure 3. After the previous analysis, we know that the value of k^* is different for different methods, which is 256, 256 and 512 for univariate, tree-based and ARR respectively. The red blocks in the heat map correspond to more overlap, while blue corresponds to less overlap.

For the univariate method, the overlap (240) of the features obtained by using the same method on the two patterns will be significantly bigger than the other overlaps. The ARR method has similar results. But for the Tree-based method, the overlap between different methods is similar. This shows that the univariate and ARR methods are more capable of selecting similar feature lists in the data sets with different load patterns.

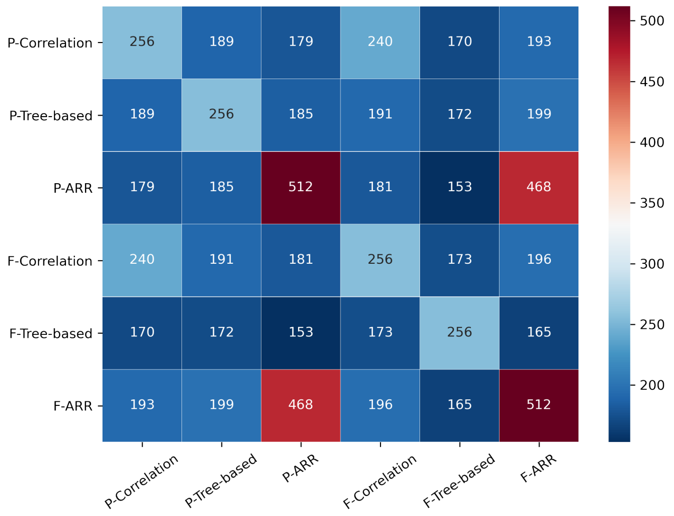


Figure 3: Heat map of the features overlap numbers in 6 (trace,method) combinations

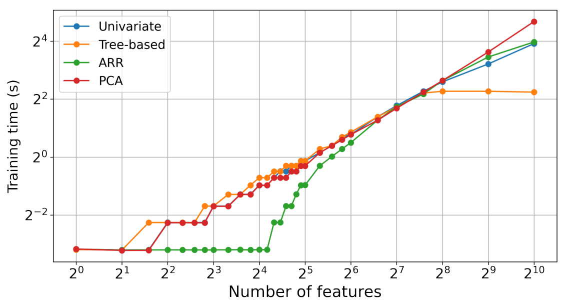
1.3 Computational overhead of dimensionality reduction and model training

The computing time for the "$ReadsAvg$" estimation in KV periodic dataset is shown in Table 3. The list in PCA represents different computing time for k = 2, 4, 8, ... The computing time for the univariate method is the shortest while the ARR method is most time-consuming.

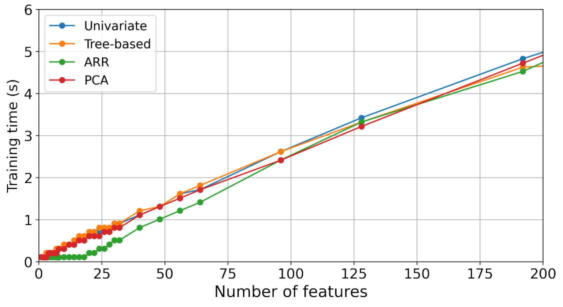
Dimensionality reduction method	**Computing time (s)**
Univariate	0.3
Tree-based	7.2
ARR	121.7
PCA	[1.0, 1.1, 1.0, 1.0 1.2, 1.4, 2.0, 3.1, 3.7, 8.8, 18.1]

Table 3: Computing time of different dimensionality reduction methods for KV Periodic

The training time using random forest regression is shown in Figure 4. In order to make the curve more accurate and intuitive, we draw the graphs of logarithmic and non-logarithmic coordinates, and calculate more points. Theoretically, the computational complexity of random forest regression training in the function of the number of features k, number of training samples m, and the number of trees n_{trees} is $O(m^2 k n_{tree})$ [6]. So the training time and k should be linear. The line corresponding to PCA proves this well, but we observe that when k is less than 200, the linear relationship between all training time and k is very good, but when it exceeds 200, the training time of univariate, Tree-based, and ARR methods are lower than expected, probably because the principal components provided by PCA are linearly independent, while other methods use correlated raw features which makes training faster. In fact, the execution of the sklearn library varies a lot from the theory and is affected by many other factors. [6] gives the algorithm complexity of various machine learning algorithms in real execution tests.



(a) Plot in log scale



(b) Plot when k is less than 200

Figure 4: Random forest regression training time with increasing number of features or principle components

Task II - Feature selection for different load patterns

The objective of this task is to design a dimensionality-reduction method that takes two traces with different load patterns of the same service as the input and produce feature sets of size k that perform equally well for different load patterns.

Basically, there are two ways to establish the new feature list and are shown in Figure 5. The first method is to first perform the dimensionality-reduction method on two load patterns (flashcrowd or periodic) respectively, get the intersection of two lists, and then sort the features based on the sum of indexes in the previous two lists. The second method is to perform the reduction method on the concatenated traces, which contains 16941 samples. The number of features and samples are shown in Table 4. In this task, we choose 1713 shared features of flashcrowd and periodic traces. We call the new feature lists obtained by the two methods integrated list and concatenated list respectively.

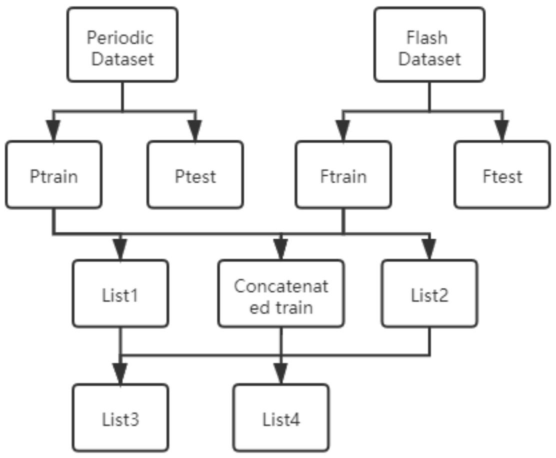
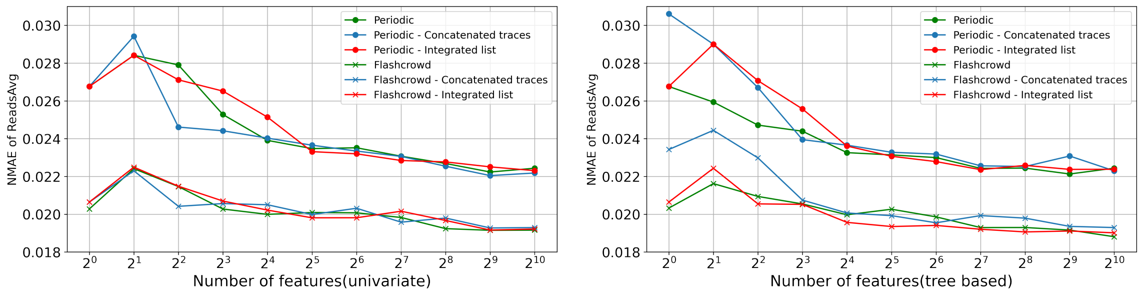


Figure 5: Two ways to establish the new feature list

	Feature	Train	Test
Flashcrowd	1723	6805	2917
Periodic	1751	10136	4345
Concatenated	1713	16941	/

Table 4: Number of features and samples of flashcrowd, periodic and the concatenated traces

Then, we evaluate the methods with respect to the accuracy of the predictor on both traces in function of k. The results are shown in Figure 6. We use the feature lists generated by the two better methods (univariate and tree based) in Task I and obtain two plots of NMAE of "$ReadsAvg$" with increasing number of features. There are six lines in each plot, corresponding to the test results of the three feature lists (the original list, integrated list, and concatenated list) in the two load pattern test sets. We can see that as the number of features increases, the difference between the three methods gradually decreases, and sometimes the results of the new feature lists we generate are even better than the original list. This shows that our two algorithms have produced new feature lists that can be used to achieve the same accuracy on both test sets as the separately trained lists.



(a) Based on features selected by the univariate method (b) Based on features selected by the tree based method

Figure 6: Random forest regression NMAE with increasing number of features using different feature lists

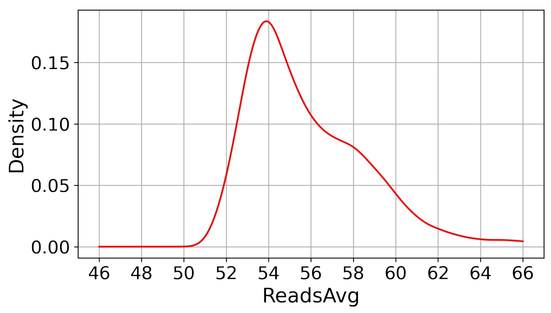
Task III - Dimensionality reduction and visualization

The objective of this task is to find a mapping of the samples of a trace into a two-dimensional space, so that the data visually separates with respect to the target values and we can better analyze the change of traces with respect to time.

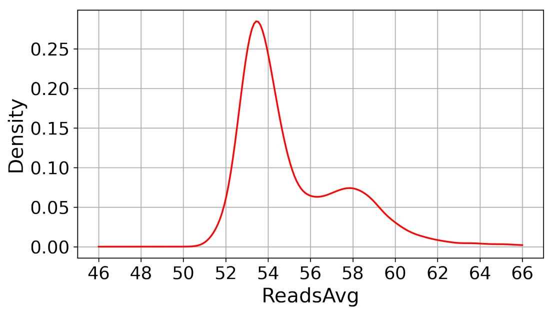
3.1 Data separation on the two-dimensional space

The density plot of "$ReadsAvg$" is in Figure 7. We can clearly see that there are two peaks in Figure 7(b), and we can use the boundary between the 2 peaks (around 56) as the classification threshold. But in Figure 7(a), the boundary point is not that obvious, which is unfavorable for later classification.

In order to plot a two-dimensional scatter-plot, two features are selected based on the Task I. The selected top 2 features are shown in Table 5.



(a) KV Periodic



(b) KV Flashcrowd

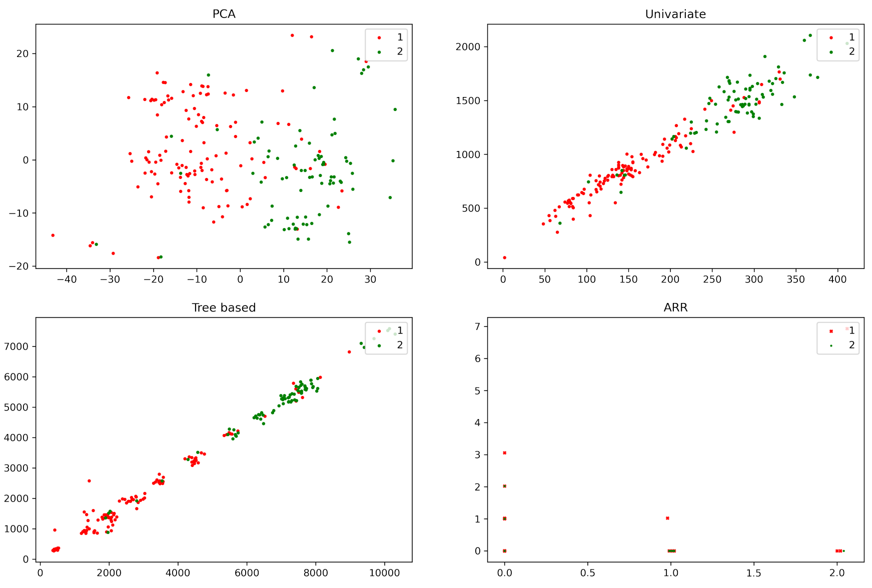
Figure 7: Density plot of target 'ReadsAvg'

Trace	Method	Feature 1	Feature 2
Periodic	Univariate	1_i127_intr.s	1_iseg.s
	Tree-based	41_RxPacktes	15_TxPacktes.1
	ARR	1_cpu6_.iowait	5_cpu14_.iowait
Flashcrowd	Univariate	0_i133_intr.s	0_i132_intr.s
	Tree-based	4_iseg.s	5_i137_intr.s
	ARR	5_cpu0_.iowait	1_cpu18_.iowait

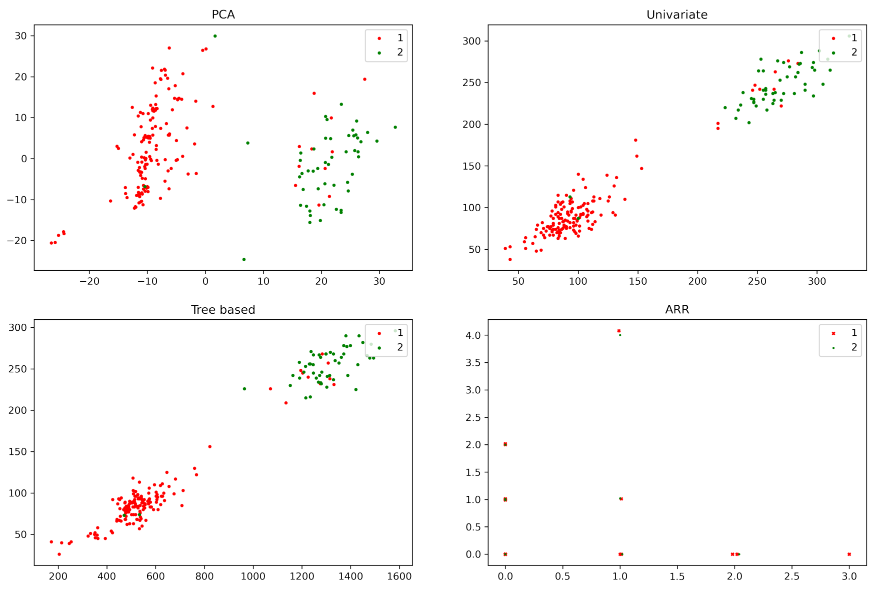
Table 5: Selected top 2 features of different methods

200 samples are selected uniformly at random to keep the plot readable. The dots are colored according to the value of the target 'ReadsAvg'. The threshold is chosen as 56 of the target value for separating the data. Samples with the target value under the threshold are colored with red and corresponds to state 1, the others are colored with green and corresponds to state 2.

The results are shown in Figure 8. The top 2 features selected from univariate and tree-based methods show clear data separation of state 1 and 2 for both traces. However, the top 2 features extracted by these two methods are highly linearly related, which shows that we may be able to complete classification with one dimension. The classification effect of PCA is not very good, but the points in the coordinate system occupy the entire space more evenly, which shows that the extracted features are more irrelevant, which may contain more information.The classification result of ARR is very poor, because the data matrix corresponding to the top 2 features is very sparse, where most of them are zero points or concentrated on certain special values.



(a) KV Periodic



(b) KV Flashcrowd

Figure 8: Two-dimensional scatter plot of selected 2 top features or principle components

The classification result of KV periodic is not as good as that of KV flashcrowd. We can verify whether it is caused by the unclear boundary of the 2 peaks in the probability density graph which was just analyzed. We mark the points with 'ReadsAvg' between 55 and 57 in orange, and the result is shown in Figure 9. We can see that the orange points are mixed between the red and green points. If the orange points are removed, then the classification result will be much better, which verifies our theory.

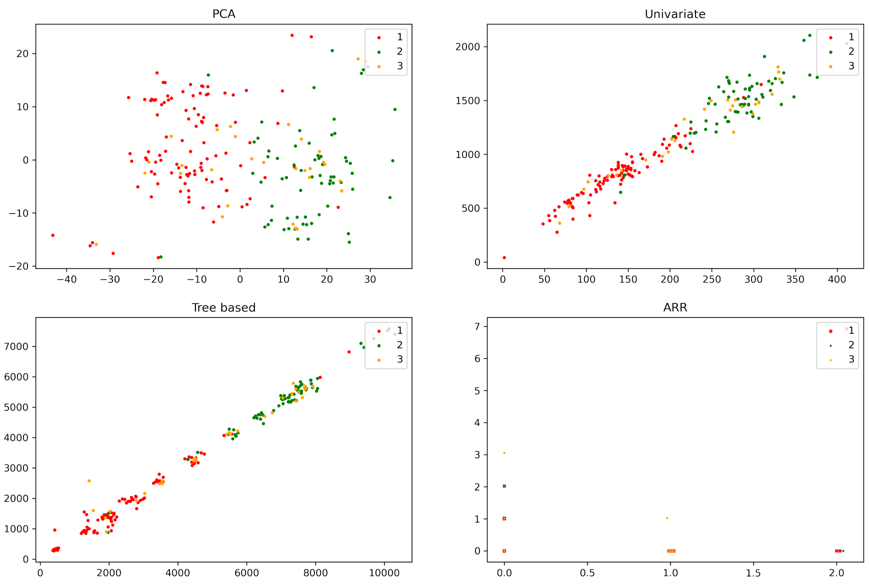
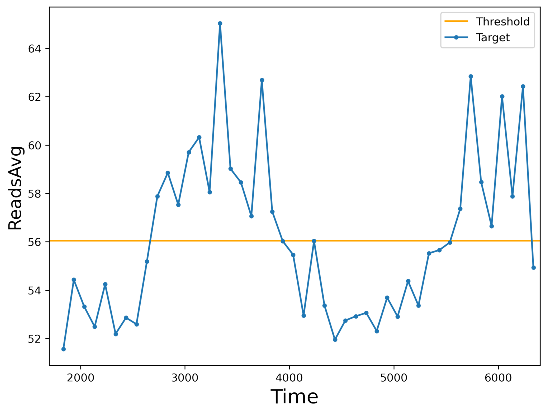


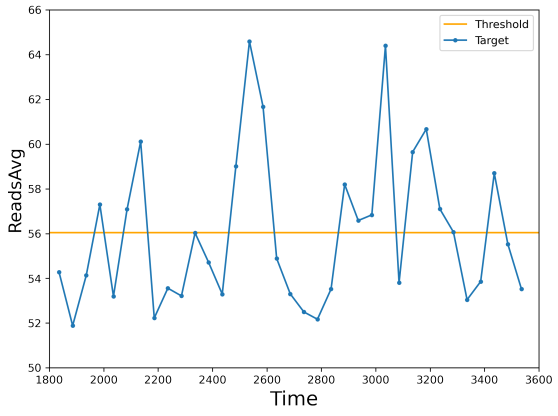
Figure 9: Two-dimensional scatter plot with 3 states for KV Periodic

3.2 The evolution of the system state

In this task, we use the top 2 principle components generated by the PCA method for better data separation. Figure 10 shows the time series plots for both traces, two system states are separated by the threshold 56. The Y axis represents the index of samples. For KV periodic, data points are chosen from the 1836th to 6375th sample at the interval of 100 samples; for KV flashcrowd, data points are chosen from the 1836th to 3540th sample at the interval of 50 samples. The sample interval chosen for KV periodic is longer because one time period of "$ReadsAvg$" is longer than that in flashcrowd. In order to observe a clearer evolution, we choose 2 hours for periodic. Figure 11 shows the evolution of the system states during the execution of the service. In 11(a), we choose every 240 samples which is roughly 4 minutes interval, in 11(b), the interval of points is 2 minutes. We can observe from the figure that the system will switch between two states, and stay for a certain period at each state.

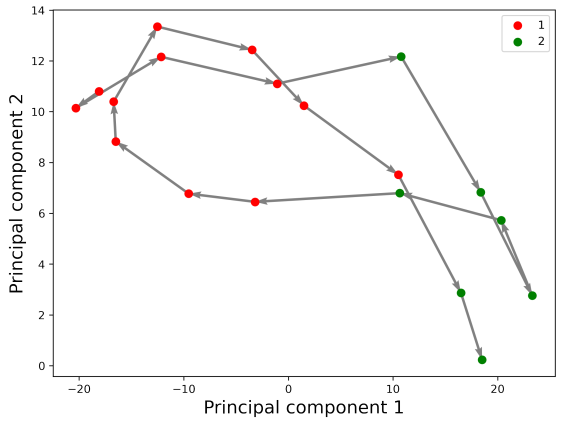


(a) 2 hours of the target value "$ReadsAvg$" in KV periodic

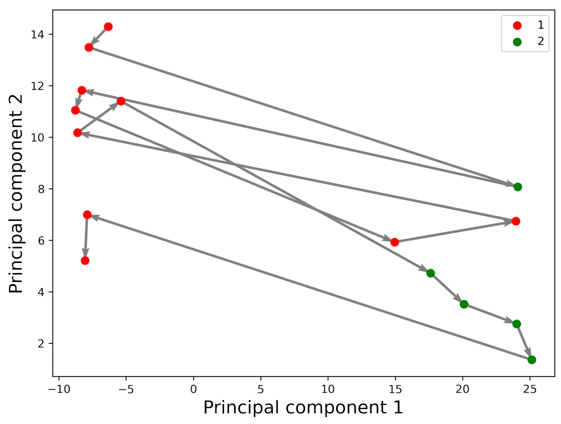


(b) 1 hour of the target value "$ReadsAvg$" in KV flashcrowd

Figure 10: Time series plot of a segment of traces



(a) KV Periodic



(b) KV Flashcrowd

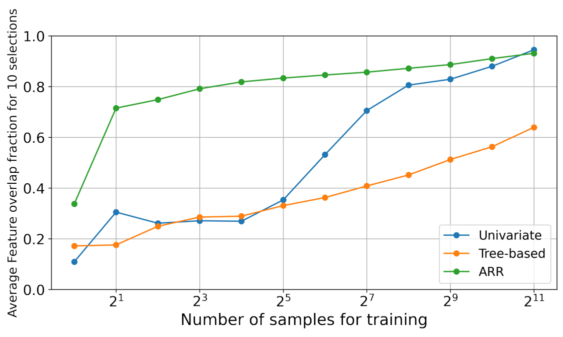
Figure 11: System state plot

Task IV - Online feature selection

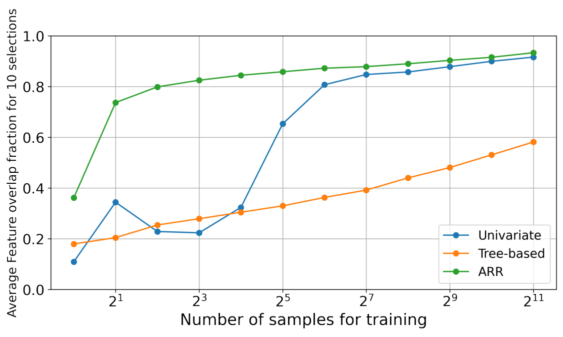
The objective of this task is to explore the changes of features calculated by feature selection algorithms over time and the subsequent impact on predictions.The samples (x_t, y_t) $t = 1, 2, 3, ...$ are ordered according to their timestamps and are processed one by one, starting with the sample with the earliest time stamp(x_1, y_1).

4.1 Online feature selection design and result

We firstly study how the top k features change over time. Following Task I, we have identified the number k^* and corresponding feature lists generated by each algorithm selection method, which can be used as a reference for later online feature selections. So we use the training sets that were used by Task I and select the subset of data in timestamp order to re-select features using the same three methods. For $t = 8, 16, 32, ..., 2048$, we calculate the fraction of features in the k^* feature set that are among the top k^* features computed in Task I. We perform this for 10 times by selecting 10 different starting points for $t = 1$ uniformly at random on the trace to eliminate the influence of the starting points of time on the experiment. The result is shown in Figure 12. For the two load pattern data sets, the results are similar. The ARR method can achieve a high list overlap fraction when the number of samples is small. The univariate method achieves a low fraction when the number of samples is small, but the fraction level will come to a high lever when the number of samples exceeds 127. For the tree-based method, the fraction value is slowly increasing, but is still vey low compared to the other two methods.



(a) KV periodic



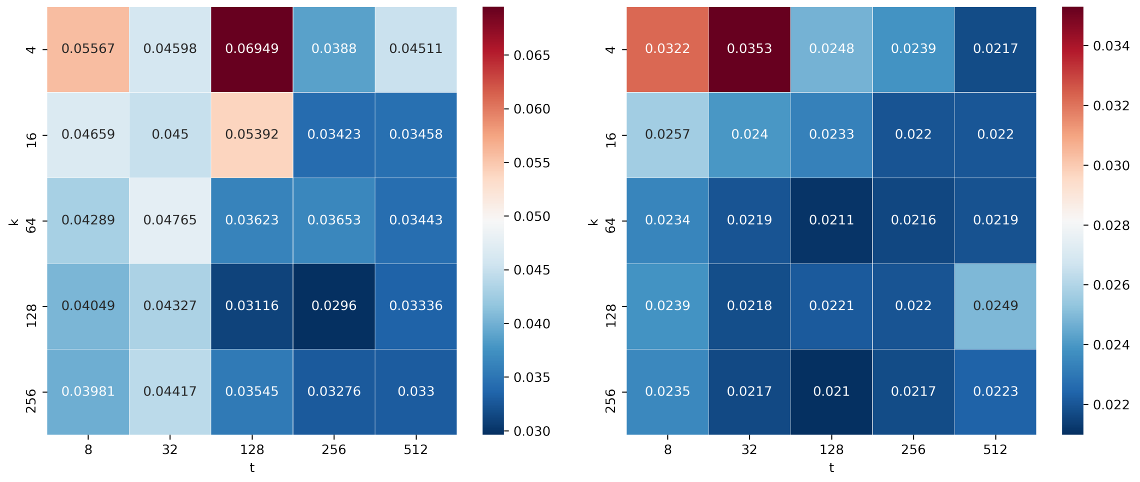
(b) KV flashcrowd

Figure 12: k^* feature fraction with increasing number of samples for training

4.2 Online feature selection impact on prediction

The tree-based feature selection method is used in this task. We utilize a 2-dimensional grid with values (k, t) where $k = 4, 16, 64, 128, 256$ and $t = 8, 32, 128, 256, 512$. For each pair (k, t), we compute the top k features using t consecutive samples from a starting point. The model is trained on the feature set determined by (k, t) and the starting point. It should be noted that we use the entire data set this time instead of the previous training set and test set.

We perform the prediction for 20 times of each (k, t) set. The starting points are selected uniformly at random on the trace. 1000 consecutive samples after a starting point are used for training and 1000 consecutive samples after the last training sample are used for testing. The heatmap of the result is shown in Figure 13. Each cell of the map corresponds to the mean accuracy of the model with respect to a (k, t) pair where blue represents low error and red represents high error. Theoretically, the training error will decrease as k and t increase respectively, because more data and features will increase the generalization ability of the model. But the growth is not endless, it will fluctuate around a certain level which is affected by various noises. It is also obviously confirmed in the heatmap. The cells in the lower right corner are overall bluer, corresponding to the lower error, and the bluest cell does not appear with the max k and t. Interestingly, on KV flashcrowd, we can achieve a test accuracy close to that of Task I, but for KV periodic, the error we get is higher than the previous result. We think this is due to the different Poisson process configurations in different load patterns. In KV periodic, we cannot use only a small part of the data to predict the subsequent data, and at the same time achieve the training accuracy of the scenario where the entire data set is randomly divided into the training set and the test set. The significance of this task is that our data sets in real engineering scenario are often very large and comprehensive. We previously considered reducing the complexity of the data sets from the aspect of selecting features, but on the other hand, we can also use a data in a smaller range of time to achieve almost the same accuracy as using the entire data set for training. From these two perspectives at the same time, we can simplify the research process and reduce engineering costs.



(a) KV periodic (b) KV flashcrowd

Figure 13: NMAE of tree-based feature selection method with (k,t) pair

Conclusion

In this project, we investigate and compare different methods with the objective to reduce the dimensionality of the feature space. We not only reduce the complexity of the model by combining dimensionality reduction and training with data over a short period of time, but also use the dimensionality reduction results to visualize the state of the system. More importantly, we verified the Manifold Hypothesis in the network system, which states that data coming from real scenario tends to occupy a small subspace of the total feature space[7]. This shows that we can use similar methods to study other systems and data sets in the network system, which is very promising.

References

[1] G. Bonaccorso, *Machine learning algorithms.* Packt Publishing Ltd, 2017.

[2] F. S. Samani, H. Zhang, and R. Stadler, "Efficient learning on high-dimensional operational data," in *2019 15th International Conference on Network and Service Management (CNSM).* IEEE, 2019, pp. 1–9.

[3] R. Stadler, R. Pasquini, and V. Fodor, "Learning from network device statistics," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 672–698, 2017.

[4] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. Long, "Managing flash crowds on the internet," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.* IEEE, 2003, pp. 246–249.

[5] X. Wang, F. S. Samani, and R. Stadler, "Online feature selection for rapid, low-overhead learning in networked systems," in *2020 16th International Conference on Network and Service Management (CNSM).* IEEE, 2020, pp. 1–7.

[6] Computational complexity of machine learning algorithms. [Online]. Available: https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1, no. 2.