UNIVERSITY OF CALIFORNIA
RIVERSIDE

Learning Based Resource Management in Mobile Cloud Computing

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Mechanical Engineering
by

Ji Guo

December 2016

Thesis Committee:
Dr. Shaolei Ren, Co-Chairperson
Dr. Marko Princevac, Co-Chairperson
Dr. Richard Wilson

The Thesis of Ji Guo is approved:

_____

_____

Committee Co-Chairperson

_____

Committee Co-Chairperson

University of California, Riverside

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Shaolei Ren, who has been a truly exceptional mentor and helped me a lot in my study at UCR. I would also like to give my thanks to Prof. Marko Princevac and Prof. Richard Wilson who greatly enriched my work using their knowledge.

I also want to give my thanks to my friend Mohammad Atiqul Islam for helping me with my thesis. I also would like to thank my parents, it was them who gave me supporting and encouragement in my living and studying at UCR

I also would like to give my thanks to my friends Qiushi Liu, Beixin Yin, Kai Li, Qi Chen, Lichao Li, Xuezhi Ma, it was them who have encouraged me in every single day.

Finally, I would like to give my thanks to Mechanical Engineering administrative staff Paul Talevera for helping me in my studying at UCR.

DEDICATION

This work is dedicated to my parents Xianhuai Guo and Baiyu Zhang, I love them forever.

ABSTRACT OF THE THESIS

Learning Based Resource Management in Mobile Cloud Computing

by

Ji Guo

Master of Science, Graduate Program in Mechanical Engineering
University of California, Riverside, December 2016
Dr. Shaolei Ren, Chairperson
Dr. Marko Princevac, Chairperson

In recent years, mobile devices such as smartphones and tablets have been playing inceasingly important roles in our life. Due to resource limitations such as storage capacity, battery lifetime, and processing ability, a smart mobile device consumes more energy than before. To address these limitations, Mobile Cloud Computing becomes a promising approach. The core of Mobile Cloud Computing is computational offloading. This thesis aims to reduce energy consumption based on the trade-off between total energy consumption and total delay time cost with the help of Support Vector Machine (SVM). In order to make a better offloading decision, this thesis chooses different key features and optimize the parameters of various kernel functions by SVM method to create a smart offloading mechanism for a mobile device. Then, it investigates three cases and shows that the total energy consumption can be reduced with the help of SVM.

Keywords: Mobile Cloud Computing, Computational Offloading, Support Vector Machine (SVM).

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction and Related Works.

1.1 Background and Motivation

In recent years, mobile devices such as smartphones and tablets have been playing increasingly important roles in our life. Due to resource limitations such as storage capacity, battery lifetime, and processing ability, a smart mobile device consumes more energy than before, and people use it more frequently than before. The battery also drains fast every day and people have to recharge it more frequently. To address these limitations, there are four approaches to reduce energy consumption of a mobile device: (1) adopting the transistors with better performance; (2) avoiding wasting energy; (3) executing slowly; (4) avoiding local computation [1] [2]. Mobile Cloud Computing provides a promising solution based on the last approach such as Amazon EC2 [3], or Microsoft Azure [4].

Due to the limitation of the resource of a mobile device, Mobile Cloud Computing can reduce the computation workload of a mobile device by computational offloading, which means sending off a specific part of the workload to a remote computer or a more resourceful server. This remote computer can be data center cluster or a cloudlet such as a mobile server. There are a significant number of researchers focusing on applications where computational offloading can be used to enhance the performance. Xiaohui Gu et al. have proposed an approach based on different kinds of applications and different resource-constrained devices, including text editing and image editing, and they have successfully reduced the execution overhead in the range of 30% to 43% [5] [6]. Zhiyuan Li et al. have considered multimedia processing on mobile devices and presented a scheme

to allocate the processing responsibility [7]. Ulrich Kremer et al. have generated two different applications of face detection and recognition, then have executed them on a mobile device and the remote server [8]. Jorge Luzuriaga et al. have applied several different algorithms into face detection, recognition, and training [9]. They also have offloaded different parts of a task to measure the different time costs and network latencies. They finally have optimized the recognition accuracy and done the trade-off between the time cost and delay latency.

1.2 Challenge of Computational Offloading

An important challenge of computational offloading is how to decide which application to offload. Many researchers are focusing on offloading schemes. Jaya Ashok Suradkar et al. have demonstrated the computational process, and the most important step is the program partitioning followed by the decision making [10]. Program partition means to identify the parts that are to be offloaded. There are various algorithms proposed in a recent research such as [11], Roopali et al. have presented that partition of a application is to divide the application into different modules and classes, and the program partition algorithm can be chosen based on different factors, or based on different optimization goals such as execution time or energy consumption. Partition process can be classified as static partition and dynamic partition, the most common is the dynamic partition. There are more studies focusing on making offloading decisions. Majid Altamimi et al. have extended previous studies and created a mathematical model to simulate different offloading situations and made the right decision based on energy consumption in communication activities [12]. Mohammed A. Hassan et al. have proposed a new framework called POMAC to enable a

task's dynamic offloading and they have pointed out that offloading is only worthwhile when the energy consumed by local execution is larger than the energy consumed by offloading. And it is pointed out that the limitation of pre-defined threshold value method which is only suited for static environment [13]. Mohammed A. Hassan et al have also selected different features such as bandwidth, latency, and data size, and they applied different classifiers such as Linear Regression, Multi-Layer Perception (MLP), SVM, and Decision Tree to make offloading decisions. They finally found that MLP, SVM, or Decision Tree with higher dimensions can provide better accuracies [13].

1.3 Support Vector Machine and Other Classifiers

Nowadays, SVM has been widely studied and applied to different areas such as handwriting recognition, facial recognition, computer vision, price prediction or even bankruptcy prediction. As for one of the classifiers, there are many types of research focusing on different kinds of classifiers' implementations and optimizations. Paulo Gaspar et al. have pointed out that compared to SVM, both KNN and NaiveBayes are pretty easy to understand, but SVM is more powerful in addressing the non-linear problem so that the SVM is the most popular algorithm due to more realistic cases are non-linear [14].

One big challenge for SVM classifier is how to optimize the kernel function. Paulo Gaspar et al. have proposed that there are 6 kinds of kernel functions and different parameters to be optimized for each function. The most common kernel functions are Polynomial kernel function and Radial Basis Function (Gaussian kernel function) [14]. According to Paulo Gaspar [14] and Chih-Wei Hsu [15], RBF (Gaussian kernel function) can provide a better

performance. Paulo Gaspar et al. have also pointed out that there are two optimization parameters for RBF kernel function and Polynomial kernel function [14].

1.4 Goals of the Thesis

The goal of this thesis is to minimize energy consumption of a mobile device based on the trade-off between total energy consumption and total delay time cost. There are two steps for this goal, the first one is to make a mobile device smarter by creating a better mechanism to make a better offloading decision. The second one is to reduce energy consumption based on the trade-off between total energy consumption and total delay time cost.

The rest of the thesis is organized as follows. Chapter 2 is for the first step. In chapter 2, I combine the idea of Ajid Altamimi [12], Aref Abdrabou [13], and Paulo Gaspar [14] to select different key features including two features and three features, then I create the energy consumption model of a mobile device. Finally, I simulate the decision-making process by using different SVM kernel functions.

Chapter 3 is for the second step. I investigate three cases to show the energy consumption can be reduced based on the trade-off between total energy consumption and total delay time cost with the help of SVM. In the first case, I select 50 groups of the total dataset to calculate the total energy consumption and the total time cost in the situation with no users' preferences. In the second case, I calculate different average energy consumptions, delay time cost rates, and energy saving rates based on different data transferred rates to show the effect of the network delay time. In the third case, I calculate different average energy

consumptions, average time costs, and average energy saving rates based on different users' preferences (offloading rates) and the trade-off between total energy consumption and total delay time cost.

Chapter 2. Different Features to Make the Decision

This chapter aims to finish the first step to create a smart mechanism for a mobile device to make a better offloading decision with the help of SVM. This chapter shows the performance of SVM based on different features and different kernel function selection. In this thesis, I take a similar application as the image processing in the research proposed by Jorge Luzuriaga [9], and the key features can be selected as workload data size, bandwidth, etc. In this chapter, I begin with testing the SVM prediction accuracies based on one situation with two different features and another with three different features. I also apply an online learning method and LIBSVM software. Then I optimize the key parameters and test the prediction accuracies again. After comparison, the result shows that more features lead to a better prediction.

2.1. Two Features

In this situation, the workload data size of the whole application of image processing and the bandwidth of the network environment can be selected as key features. For the workload data size, I assume that there are different kinds of execution tasks with different workload data sizes ranging from 0 to 10MB.

For the bandwidth, I only consider about WIFI environment with the value range from 0 MB/s to 30 MB/s. Two-feature situation is shown in Table 1.

Table 1. Two Features

Feature	Representation	Value
Workload	Total data size of computation tasks including the whole size of image and other applications	0 - 10 MB
Bandwidth	Data transmission speed	0 - 30 MB/s

2.2 Decision Function

Decision function $f(x, y)$ is to create the original training set based on the energy consumption of mobile device in two situations: local computing and offloading computing. For task μ_i with the feature (x_i, y_i), if $f(x_i, y_i) > 0$, it means the task is suitable for offloading, and it is labeled as 1; if it is less than 0, it is labeled as -1 which means it is not suitable for offloading. Table 2 shows different parameters of decision function.

Table 2. Parameters of Decision Function

Symbol	Representation	Value
μ_m	Utilization of the mobile device	80%
p_{pa}	Peak power of the mobile device in active state	2W
p_d	Power of the mobile device in idle state	0.4W
S_{mp}	Processing speed of the mobile device	15MB/s
P_{tr}	Transmission power of offloading computing	0.6W
x	Workload data size	0 - 10 MB
y	Bandwidth	0 - 30 MB/s

2.21 Local Computing

In the local computing situation, the total energy consumption of the mobile device is the energy model in [1] as:

$$f_{local}(x) = \left(\mu_m \cdot p_{pa} + p_d\right) \cdot \frac{x}{S_{mp}} \quad (1)$$

Where x represents the workload data size.

2.22 Offloading Computing

In the offloading computing situation, the energy consumption of the mobile device is caused by data transmission. For the transmission energy consumption, I apply the model in [1] as function (2):

$$f_{offloading}(x, y) = \mathrm{P_{tr}} \cdot \frac{x}{y} \quad (2)$$

Here, x is the workload data size, and y is the bandwidth. The final decision function is the difference of two parts as function (3):

$$f(x, y) = \left(\mu_\mathrm{m} \cdot p_{pa} + p_d\right) \cdot \frac{x}{S_{mp}} - \mathrm{P_{tr}} \cdot \frac{x}{y} \quad (3)$$

2.3 Three Features

In this situation, the third new feature is the data transferred as shown in Table 3.

Table 3. Three Features

Feature names	Representation	Value
Workload	Total data size of a task	0 - 10MB
Data transferred	Offloading part of the total workload data size	0 - 10MB
Bandwidth	Data transmission speed	0 - 30MB/s

2.4 Decision Function

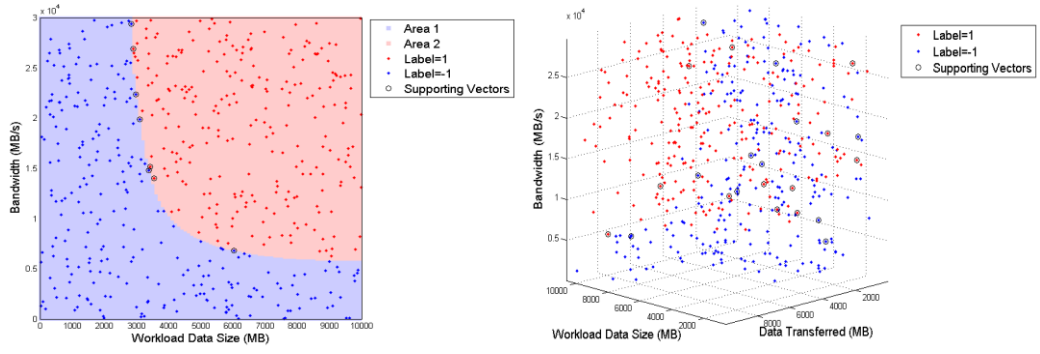
For three features, the decision function is as function (4):

$$f(x,y,z) = (\mu_m \cdot p_{pa} + p_d) \cdot \frac{x}{s_{mp}} - (\mu'_m \cdot p_{pa} + p_d) \cdot \frac{(x-z)}{s_{mp}} - P_{tr} \cdot \frac{z}{y} \quad (4)$$

Where z represents the data transferred and μ'_m represents the utilization of the mobile device in offloading state which is 50%.

2.5 Dataset Distribution and Classification

To simulate a real computation task, according to function (3) and (4), two 500-group datasets with two features and three features which are around 50% labeled as 1 and around 50% labeled as -1 are created. In the two-feature situation, for task μ_i with the feature (x_i, y_i), the total dataset is a 500×3 matrix with the first column as the labels. In the three-feature situation, for task μ'_i with the feature (x'_i, y'_i, z'_i), the total dataset is a 500×4 matrix with the first column as the labels. The dataset distribution and classification is shown in Fig.1.



(a) (b)

Fig.1 Dataset distribution and classification. (a) Two-feature situation. (b) Three-feature situation.

2.6 Prediction Accuracy Test Based on Online Learning Method

2.6.1 Online Learning Method

In this section, I apply the software, LIBSVM, to test prediction accuracy based on the SVM method. For the online learning method, the first two-group of tasks with different labels are applied to be a training set to create an original training model. After the third task arrives, LIBSVM directly puts the new task into the predicting set and predicts it to create a new label based on the previous model, if the predicted label is same with the original label, the prediction is correct and the accuracy will be 100%, if not, it is not a correct prediction and the accuracy will be 0. Then LIBSVM puts the three tasks together to create a new training set, then it predicts the new one and it keeps repeating the process until finishing the total dataset. The prediction accuracy of each round equals the ratio of the number of correct prediction and the total number of prediction.

2.6.2 Online Learning Method Based on Two Kernel Functions

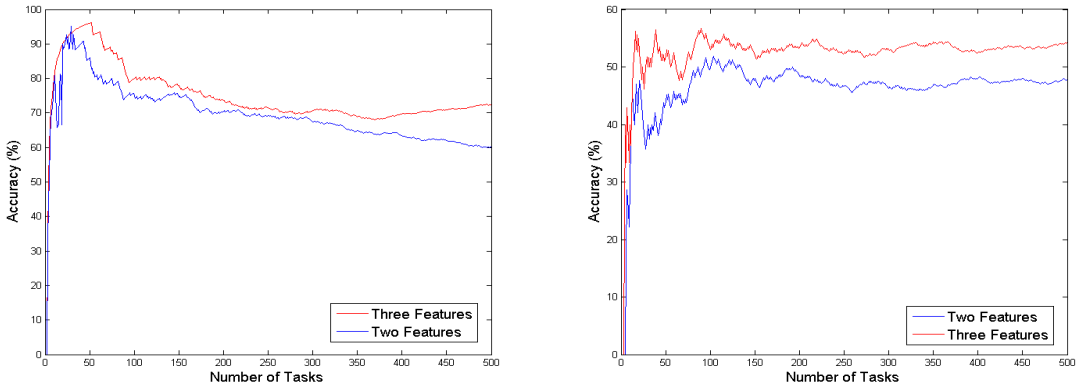
Fig.1 shows that two original datasets are non-linear separable, then the kernel function method has to be considered. In this thesis, two most common kernel functions are applied to test the prediction accuracy: Polynomial kernel function (5) and Gaussian kernel function (6).

$$k\,(\,x,y\,) = (\langle x_1, x_2\rangle + R)^d \tag{5}$$

$$k\,(\,x,y\,) = \; e^{(-\gamma\,\|x-y\|^2)}\;, \gamma > 0 \tag{6}$$

Fig.2a shows the prediction accuracies of the two-feature situation and the three-feature situation based on a Polynomial kernel function. For the Polynomial kernel function, the

accuracy is unstable. Fig.2b shows the prediction accuracy based on the Gaussian kernel function. For the Gaussian kernel function, the final result is more stable compared with that of the Polynomial kernel function. Results show that for the two-feature situation, the prediction accuracy is only around 48%, but for the three-feature situation, the prediction accuracy is around 53%. To improve the prediction performance, some parameters need to be optimized in the next part.



(a) (b)

Fig.2 Prediction accuracies of online learning based on different kernel functions without optimization. (a) Polynomial kernel function. (b) Gaussian kernel function

2.7 Cross Validation and Parameter Optimization

2.7.1 Cross Validation

In this part, a 5-fold cross validation method is applied to find out the optimal value of different parameters to improve accuracy.

2.7.2 Parameters Optimization of two Kernel Functions.

According to Nikodin Ristanovic [19], SVM classification problem is as function (7):

$$\min_{w,b,\varepsilon} \ \frac{1}{2} \cdot w^T w + C \sum_{i=1}^{l} \varepsilon_i \tag{7}$$

According to Paulo Gaspar [14] and Ben-Hur [16] , for a Gaussian kernel function, two parameters need to be optimized, C and kernel function's γ where C is the parameter for the soft margin cost function, which trades off misclassification of training set. Here, if C has a larger value, it will lead to a narrow margin; if γ has a smaller value, it will lead to a more linear decision boundary. For a Polynomial kernel function, C and degree (d) need to be optimized, higher degree (d) will lead to a more flexible decision boundary, and larger C will lead to a narrow margin.

Table 4 shows the value range of different parameters to be optimized. According to Chih-Wei [15], γ and C can be calculated in exponentially growing sequences($C = 2^x, \gamma = 2^y$). Then the cross-validation method is used to calculate the best cross-validation prediction accuracy and obtain the optimal value. In [15], Chih Wei Hsu et al. have regularly set the value range of C and γ as $2^{-5} \sim 2^5$. Here, I set the value range of C and γ as $2^{-50} \sim 2^{50}$ in order to get more general results. After optimization, in the two-feature situation, the best cross-validation prediction accuracy provided by the Polynomial kernel function is 98.6%, the best cross-validation prediction accuracy provided by the Gaussian kernel function is 99.2%. In the three-feature situation, the best cross-validation prediction accuracy provided by the Polynomial kernel function is 98.8%, the best cross-validation prediction accuracy provided by the Gaussian kernel function is 99.4%.

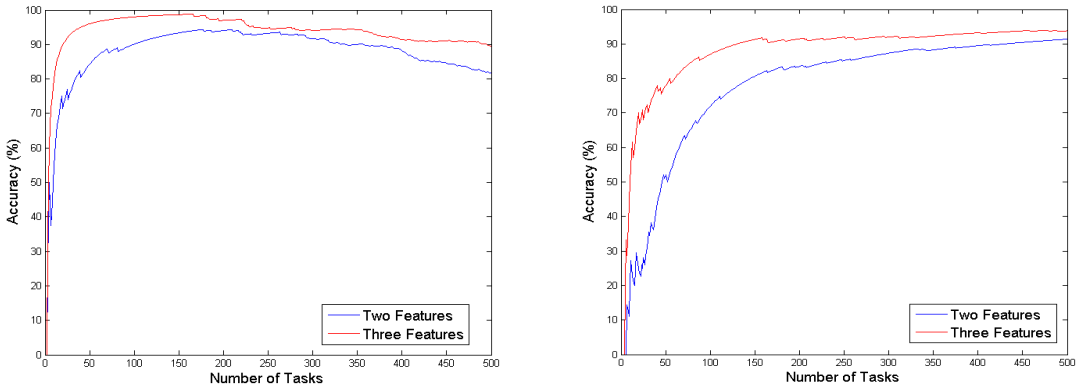
Table 4. Parameters Selection and Optimization

Parameters	Polynomial (1)	Gaussian (2)	Value Range	Optimal Value in Two Features		Optimal Value in Three Features	
				(1)	(2)	(1)	(2)
C	✓	✓	$2^{-50} \sim 2^{50}$	2^{15}	2^{14}	2^{22}	2^{25}
γ		✓	$2^{-50} \sim 2^{50}$		2^{-29}		2^{-37}
d	✓		$1 \sim 50$	4		7	

2.8 Prediction Accuracy Test after Optimization

Fig.3a shows the online learning accuracy based on the Polynomial kernel function with optimization. Results of the three-feature situation are better than the results of the two-feature situation, but the results provided by the Polynomial kernel function is not stable, as the number of tasks increases, the accuracy increases firstly and begins to decrease.

Fig.3b shows online learning accuracies based on the Gaussian kernel function. The accuracy result in the three-feature situation is also better than the accuracy in the two-feature situation, but the overall result provided by the Gaussian kernel function is better than the Polynomial kernel function. As a number of tasks increases, accuracies become more and more stable and reach to around 91% and 93%.



(a) (b)

Fig.3 Prediction accuracies of online learning based on different kernel functions without optimization. (a) Polynomial kernel function. (b) Gaussian kernel function. (c) Comparison of the prediction accuracies based on different classifiers.

2.9 Comparison with Different Classifiers

There are many researchers focusing on prediction based on different classifiers. E. Cuervo et al. have proposed a system, MAUI, to improve the application partition based on a Linear Regression model which is also used to capture the relationship between system features. But the result shows that the features cannot capture more features in the research proposed by E.Cuervo [17]. According to current research proposed by Mohammed [13], the accuracy of a Linear Regression model is only around 43%. It also shows that a Threshold model will lead to a 50% wrong prediction but for a SVM model, it can only lead to a wrong prediction which is less than 15%. It also shows higher dimensional classifiers such as Multi-Layer Perception (MLP), SVM, and Decision Tree can achieve a better accuracy which is higher than 80%. Fig.4 shows the comparison of the prediction accuracies of different classifiers including Threshold, Linear Regression, NaiveBayes, Decision Tree

and MLP with the results obtained from Mohammed [13] and E.Cuervo [17]. It can be seen that SVM can provide the best prediction accuracy which is over 90% after optimization.

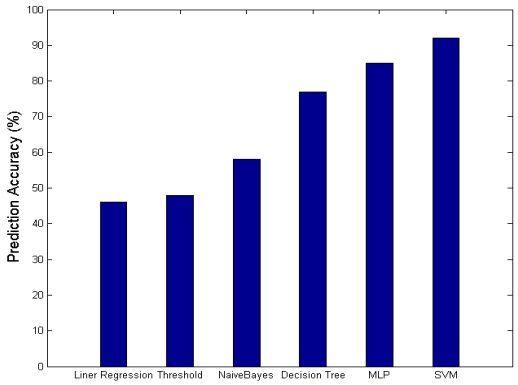


Fig.4 Comparison of the prediction accuracies based on different classifiers.

2.10 Conclusions

This chapter is aimed to finish the first step to create a smart mechanism for a mobile device to make a better offloading decision.

In this chapter, I test the prediction accuracy by an online learning method based on two different kernel functions in different key-feature situations. And I apply the cross-validation method to optimize different parameters of the different kernel functions and retest the prediction accuracy. For the Gaussian kernel function, the prediction accuracy based on the two-feature situation is around 48% without optimization, after optimization, the accuracy can reach to around 91%. In the three-feature situation, the prediction accuracy is around 53% before optimization, and the accuracy is around 93% after optimization. Result provided by the Polynomial kernel function is stable at the beginning, but as the number of tasks increases, the accuracy begins to decrease.

After comparison between two situations with different features and different kernel functions, the first step of the goal is finished. It shows that the situation with more features can lead to a better prediction accuracy, and the Gaussian kernel function can provide a better prediction than the Polynomial kernel function does

Chapter 3. Performance and Different Offloading Decisions

This chapter aims to finish the second step to reduce energy consumption based on the trade-off between the energy consumption and the total delay time cost with the help of SVM. To finish the second step, a new feature, queueing delay time is introduced, and three cases are investigated. The first case shows trends of total energy consumption and total delay time cost. The second case shows the effect of network delay time cost. The third case shows total energy consumption can be reduced based on the trade-off between energy consumption and total delay time cost with the help of SVM.

3.1 Fourth Feature: Queueing Delay Time

In the previous chapter, the two-feature situation and the three-feature situation are considered. In the offloading situation, queueing delay time is another key feature which cannot be ignored. Here, I assume the queueing delay time with the value range of 0 to 200ms. The features are shown in Table 5.

Table 5. Four Features

Feature names	Representation	Value
Workload	Total data size of a task	0 -10MB
Data transferred	Offloading part of the total workload data size	0-10MB
Bandwidth	Data transmission speed	0 - 30MB/s
Delay time	The queueing delay time	0 -200 ms

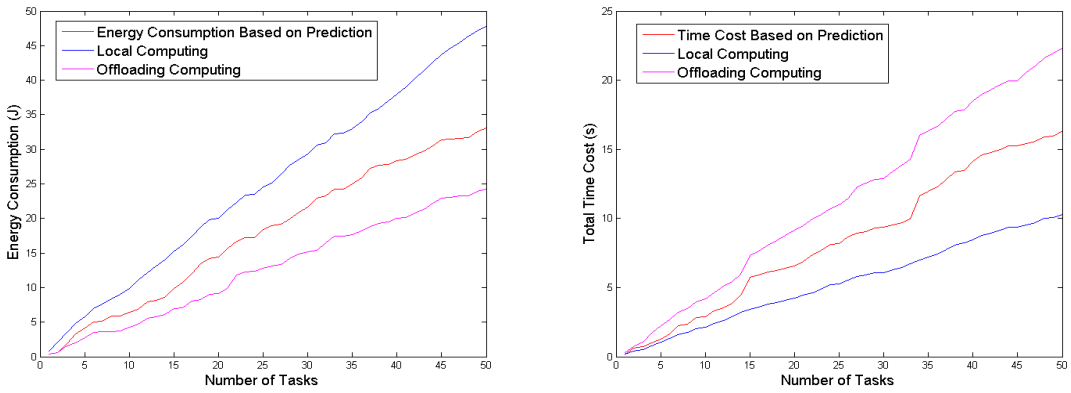
Table 6. Total Energy Consumption and Energy Consumption Caused by Delay Time

Offloading Conditions	Offloading based on Prediction		Always Offloading	No Offloading
	Offloading computing	Local computing		
Energy Consumption	$(\mu'_{\mathrm{m}} \cdot p_{pa} + p_d) \cdot (\frac{workload - data\ trasferred}{S_{mp}}) + P_{tr} \cdot \frac{data\ transferred}{bandwidth}$	$(\mu_{\mathrm{m}} \cdot p_{pa} + p_d) \cdot \frac{workload}{S_{mp}}$	$(\mu'_{\mathrm{m}} \cdot p_{pa} + p_d) \cdot (\frac{workload - data\ trasferred}{S_{mp}}) + P_{tr} \cdot \frac{data\ transferred}{bandwidth}$	$(\mu_{\mathrm{m}} \cdot p_{pa}) + p_d) \cdot \frac{workload}{S_{mp}}$
Energy Consumption due to Network Delay Time	$P_{tr} \cdot \frac{data\ transferred}{bandwidth}$	0	$P_{tr} \cdot \frac{data\ transferred}{bandwidth}$	0
Total Delay Time	$\left(\frac{workload - data\ trasferred}{S_{mp}}\right) + \frac{data\ transferred}{bandwidth} + T_{delay}$	$\frac{workload}{S_{mp}}$	$\left(\frac{workload - data\ trasferred}{S_{mp}}\right) + \frac{data\ transferred}{bandwidth} + T_{delay}$	$\frac{workload}{S_{mp}}$

The decision function of the four-feature situation is same with the function (4), the total dataset is created in a similar way as the previous chapter. In the following section, case1 is investigated to show the trends of the energy consumption and the total delay time cost; case 2 is investigated to show the effect of the network delay time; case 3 shows total energy consumption can be reduced based on the trade-off between total energy consumption and total delay time cost. The relationship between energy consumption and delay time cost is shown in Table 6.

3.2 Case 1: Trends of Total Energy Consumption and Total Delay Time Cost.

In this case, trends of the total energy consumption and the total delay time cost of 50 groups of tasks are shown in an accumulating method in Fig.4.



(a) (b)

Fig.5 Total energy consumption and total time cost. (a) Energy consumption (b) Total delay time cost

Fig.5 shows the total energy consumption based on 50 groups of tasks including the always-offloading situation, the offloading situation based on a prediction by SVM and the local computing (no offloading) situation. Fig.5a shows that the local computing situation

leads to the most energy consumption. Fig.5b shows that the total time cost based on 50 groups of tasks. According to Fig.5b, the always-offloading situation results in the most time cost. In the next section, the effect of the network delay time is investigated.

3.3 Case 2: Effect of the Network Delay Time Cost

This case shows the effect of network delay time cost based on different data transferred rates with the help of SVM. In the three-feature situation, data transferred is the feature which is assumed to be a random value, in this case, the data transferred part is assumed to be several different rates of the total workload data size, and the value range of rate is $10\% \sim 90\%$.

For total energy consumption of each task, according to the decision function (2) and (3), total energy consumption can also be written as function (8):

$$\mathrm{E}_{total} = \mathrm{E}_{processing} + \mathrm{E}_{network\_delay} \tag{8}$$

Where $\mathrm{E}_{processing}$ is the energy consumption due to the local computing. $\mathrm{E}_{network\_delay}$ is the energy consumption due to the network delay time.

Where β is the network delay cost rate of the task, which means the energy consumption caused by the network delay time divided by the total energy consumption based on prediction. β can be written as function (9):

$$\beta = \mathrm{E}_{network\_delay} \,/\, \mathrm{E}_{total} \tag{9}$$

For the energy saving rate ε based on SVM prediction, it represents the difference of energy consumption of local computing (E_{total_local}) and offloading computing based on

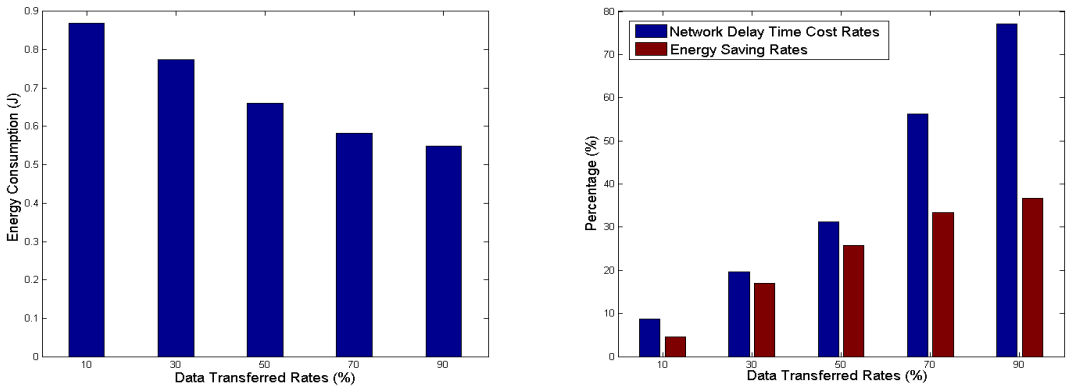
prediction ($E_{total\_prediction}$) divided by the energy consumption of local computing (E_{total_local}). It can be written as function (10)

$$\varepsilon = (E_{total_{local}} - E_{total_{prediction}}) / E_{total\_local} \quad (10)$$

Table 7 shows the parameters change based on different data transferred rates, and different energy consumption parts are calculated based on Table 6.

Table 7. Parameters Change Based on Different Data Transferred Rates

Data Transferred Rates	μ'_m
10%	70%
30%	60%
50%	50%
70%	40%
90%	30%



(a) (b)

Fig.6 Average energy consumptions, average energy saving rates, and average network delay time cost rates. (a) Average energy consumptions based on different data transferred rates. (b) Average energy saving rates, and average network delay time cost rates

Fig.6a shows different energy consumptions based on different data transferred rates. Fig.6b shows that as the data transferred rate increases, the energy consumption caused by

the network delay time cost will account a larger rate in the total energy consumption, and the network delay cost rate β is increasing. It can also be seen that the energy saving rate ε increases fast at the beginning, when the data transferred rate is larger than 50%, ε increases slower. In additionally, the delay time cost rate β increases slower at the beginning, when the data transferred rate is larger than 50%, β increases fast. This results from the energy consumption caused by the network delay time cost which becomes more and more dominant.

3.4 Case 2: Trade-off between Total Energy Consumption and Total Delay Time Cost

In the section above, the effect of the network delay time cost based on different data transferred rates is investigated. In this section, the trade-off between total energy consumption and total delay time cost is investigated in order to show total energy consumption can be reduced.

According to the decision function (4) and Table 6, for each task, it has two cost functions as function (11) and (12):

$$C_1 = \lambda_1 \mathrm{E}_{total\_local} + \lambda_2 \cdot \mathrm{T}_{total\_local} \quad (11)$$

$$C_2 = \lambda_1 \mathrm{E}_{total\_offoading} + \lambda_2 \cdot \mathrm{T}_{total\_offloading} \quad (12)$$

Where $\lambda_1(\$/\mathrm{J})$ is the energy weight factor, and λ_2 $(\$/\mathrm{s})$ is the time weight factor.

It can also be written as function (13) and (14):

$$C_{local} = \mathrm{E}_{total\_local} + \lambda \cdot \mathrm{T}_{total\_local} \quad (13)$$

$$C_{offloading} = \mathrm{E}_{total\_offoading} + \lambda \cdot \mathrm{T}_{total\_offloading} \quad (14)$$

Where $\lambda = \frac{\lambda_2}{\lambda_1}\ (J/s)$ is the ratio of weight factors. $E_{total\_offloading}$ is the total energy consumption in offloading computing situation, and E_{total_local} is the total energy consumption in local computing situation. T_{total_local} is the total delay time cost in local computing situation, and $T_{total\_offloading}$ is the total delay time cost in offloading computing situation. $C\ (J)$ is the total cost value combined with the total energy consumption and the total delay time cost.

Table 8. Cost Function Values and λ

Offloading status	Total Cost Value C (J)	λ(J/s)	Dominant Factor
Offloading More	$C_{local} > C_{offloading}$	$\lambda \ll 1$	Energy
Equally	$C_{local} \approx C_{offloading}$	$\lambda \approx 1$	Energy or Time
Offloading Less	$C_{local} < C_{offloading}$	$\lambda \gg 1$	Time

Table 8 shows the influence of λ on functions (13) and (14). As λ is larger, total time cost is more important to mobile users, then the possibility of $C_{offloading} > C_{local}$ is stronger. In this situation, for sample task μ_i with the feature $(x_i, y_i, z_i, \varphi_i)$, if $C_{offloading\_i} > C_{local\_i}$, task μ_i will be labeled as not suitable for offloading and vice versa, then a new training set is created. According to function (13) and function (14), after the prediction by the new SVM model based on the new training set, the general total cost function (15) can represent how to make the decision based on different users' preferences for the total dataset.

$$C_{total} = (1 - \delta) \cdot C'_{local} + \delta \cdot C'_{offloading} \quad (15)$$

Where δ is the offloading rate depending on λ. C'_{local} is the total cost of local computing based on the sample data with the predicted label as -1, $C'_{offloading}$ is the total cost of offloading computing based on the sample data with the predicted label as 1. C_{total} is the total cost value of the total 500-group dataset.

The second step of this case is to minimize the total cost function (14) which is the combination of total energy consumption and total delay time cost. The relationship between λ and δ is shown in Table 9. Larger λ leads to smaller δ because larger λ means total delay time cost is more important and users should offload less. According to the cost function (12) and (13), all sample data are labeled based on its cost value after comparing $C_{offloading}$ and C_{local}. For the total dataset, C_{total} can be minimized by this method.

Table 9. Different Offloading Modes

Offloading Modes	Offloading Rates (δ)	λ (J/s)
Hardly Offloading (1)	10%	10000
Lowly Offloading (2)	30%	100
Median Offloading (3)	50%	1.1
Highly Offloading (4)	70%	0.01
Almost Fully Offloading (5)	90%	0.0001

Next, various average total energy consumptions, average total delay time costs, and average energy saving rates in different offloading modes are calculated according to Table

6. After calculation, various average energy consumptions, average total time costs, and average energy saving rates of different modes are showed in fig.7.

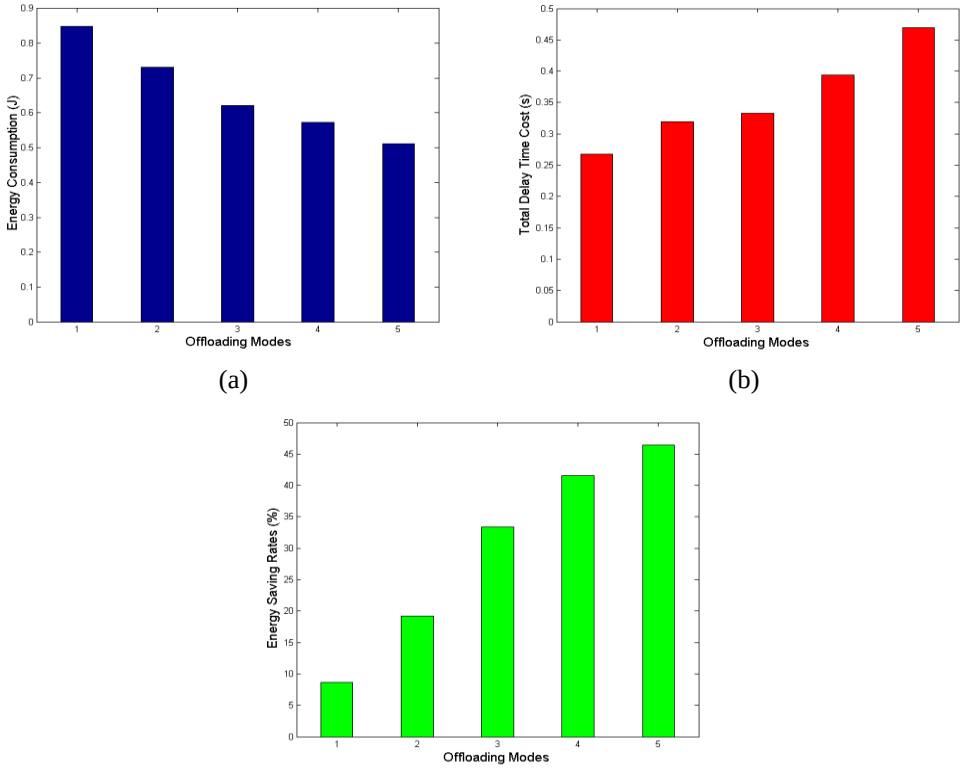


Fig.7 Average energy consumptions, average total delay time costs, and average energy saving rates based on different offloading modes; (a) Average energy consumptions based on different offloading modes; (b) Total delay time costs based on different offloading modes; (c) Energy saving rates based on different offloading modes.

Fig.7a shows various average total energy consumptions in different offloading modes. Fig.7b shows various average total delay time costs in different offloading modes. Fig.7c shows various average energy saving rates in different offloading modes.

3.5 Comparison with Related Works

For the energy saving, there are many researchers focusing on it and they have obtained many results based on different implementations. E. Cuervo et al. have applied MAUI and Linear Regression model into video games and MAUI has saved 27% energy [17]. Young-Woo Kwon et al. have proposed a new approach to reduce the energy consumption by offloading, the result shows that this approach has saved around 25% energy consumption [18]. In additionally, Nikodin Ristanovic et al. have designed two algorithms to reduce the energy consumption in different network environments. And they have saved around 30% energy in Wi-Fi environment [19]. Yeli Geng et al. have proposed a new algorithm to minimize the energy consumption and investigate the effect of tail latency in offloading situation. The algorithm has reduced the energy consumption by 35% due to the application is executed based on a better network environment LTE [20]. In this thesis, the average energy saving rate based on the SVM model can be reach to 30% in case 1. In case 2, the energy saving rate is around 27%, and it is 31% in case 3. In this section, 31% is applied to do the comparison. Fig.8 shows the comparison of different energy saving rates.

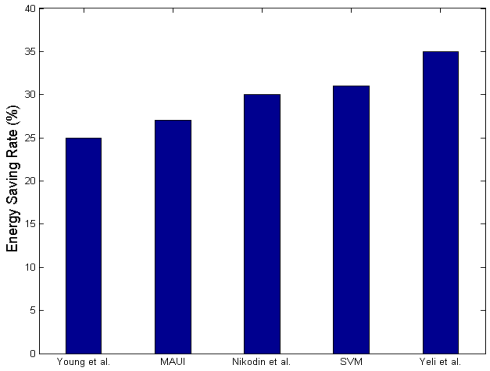


Fig.8 Comparison of different energy saving rates.

3.6 Conclusions

In this chapter, I investigate three cases to finish the second step to show total energy consumption can be reduced based on the trade-off between total energy consumption and total delay time cost with the help of SVM. Case 1 shows trends of the energy consumption and the time cost in the situation with no users' preferences, this result shows that for a groups of dataset, offloading computing leads to less total energy consumption but more total delay time cost. In case 2, the situation with different offloading data transferred rates is investigated to show the effect of the network delay time cost. Results show that higher data transferred rate leads to higher delay cost rate which means the energy consumption caused by the network delay time accounts more in the total energy consumption. In case 3, total energy consumption and total delay time cost are combined more generally, ratio of weight factors rate λ is introduced to represent mobile users' preferences. Larger λ means total delay time cost is more important to mobile users and smaller λ means total energy consumption is more important to mobile users. Next, different situations with different offloading modes are investigated based on λ where different average energy consumptions, different total delay time costs, and different energy saving rates are calculated. This chapter shows energy consumption can be reduced based on the trade-off between total energy consumption and total delay time cost with the help of SVM.

Chapter 4. Conclusions

This thesis is aimed to reduce total energy consumption of based on the trade-off between total energy consumption and total delay time cost of a mobile device. There are two steps for this goal, the first one is to create a smart mechanism for a mobile device to make a better offloading decision, and the second one is to show total energy consumption can be reduced based on the trade-off.

The first step is finished in chapter 2, the two-feature situation and the three-feature situation are considered to do the prediction accuracy test based on different kernel functions by the SVM method. And different optimal parameters are obtained by the cross-validation method based on different situations with different features and different kernel functions. The results of different accuracy show that the situation with more key features can lead to a better prediction, and the Gaussian kernel function can provide a better prediction accuracy then create a smart mechanism for a mobile device.

The second step is finished in chapter 3 where three different cases are investigated to do the trade-off between total energy consumption and total delay time cost. Case 1 shows that local computing leads to more total energy consumption but less total delay time cost; offloading computing leads to more total delay time cost but less total energy consumption. In case 2, the effect of network delay time cost is investigated based on different data transferred rates, and results show that higher data transferred rate leads to higher delay time cost rate which means the energy consumption caused by the network delay time accounts more in the total energy consumption. In case 3, the ratio of weight factor rate λ

is introduced to represent different mobile users' preferences. In this case, various total energy consumptions, total delay time costs, and different energy saving rates are calculated based on different users' preferences (offloading modes). It can be seen that total energy consumption can be reduced based on the trade-off between total energy consumption and total delay time cost with the help of SVM.

References

[1] Karthik Kumar, Yung-Hsiang Lu. Cloud Computing for Mobile Users: Can Offloading Computation Save Energy. Computer, vol.43, pp.51-56, 2010.

[2] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, Bharat Bhargava. A Survey of Computation Offloading for Mobile Systems. Mobile Networks and Applications archive, vol.18, pp.129-140, 2013.

[3] Amazon EC2 https://aws.amazon.com/ec2/

[4] Micosoft Azure https://azure.microsoft.com/en-us/

[5] Gu X, Nahrstedt K, Messer A, Greenberg I, Milojicic D. Adaptive Offloading Inference for Delivering Applications. Pervasive Computing Environments. IEEE international conference on pervasive computing and communications, pp.107–114, 2003.

[6] Xiaohui Gu , K. Nahrstedt , A. Messer, I. Greenberg. Adaptive Offloading Inference for Delivering Applications. Pervasive Computing Environments. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, vol.26, pp.107 – 114, 2003.

[7] Zhiyuan Li, Cheng Wang, Rong Xu. Task Allocation For Distributed Multimedia Processing On Wirelessly Networked Handheld Devices. Parallel and Distributed Processing Symposium. Proceedings International, pp.15-19, 2001.

[8] Ulrich Kremer, Jamey Hicks, James Rehg. A Compilation Framework for Power and Energy Management on Mobile Computers. Proceedings of the 14th international conference on Languages and compilers for parallel computing, pp.115-131, 2003.

[9] Jorge Luzuriaga, Juan Carlos Cano, Carlos Calafate, Pietro Manzoni. Evaluating Computation Offloading Trade-offs in Mobile Cloud Computing: A Sample Application. Mobile Networks and Applications, pp.1-16, 24, 2016.

[10] Jaya Ashok Suradkar, R. D. Bharati. Computation Offloading: Overview, Frameworks and Challenges. International Journal of Computer Applications, vol.134, pp 28-31, 2016.

[11] Roopali, Rajkumari. Overview of Offloading in Smart Mobile Devices for Mobile Cloud Computing. International Journal of Computer Science and Information Technologies, vol.5, pp.7855-7860, 2014.

[12] Ajid Altamimi, Atef Abdrabou, Kshirasagar Naik,and Amiya Nayak. Energy Cost Models of Smart phones for Task Offloading to the Cloud. IEEE Transactions on Emerging Topics in Computing, vol.3, pp.384 – 398, 2015.

[13] Mohammed A. Hassan1, Kshitiz Bhattarai, Qi Wei1 and Songqing Chen. POMAC: Properly Offloading Mobile Applications to Clouds. Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing, pp.7-7.

[14] Paulo Gaspar, Jaime Carbonell and Jose Lu´ıs Oliveira. On The Parameter Optimization of Support Vector Machines for Binary Classification. Journal of Integrative Bioinformatics, vol.9, pp.201, 2012.

[15] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification。

[16] Asa Ben-Hur, Jason Weston. A User's Guide to Support Vector Machines

[17] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman,S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making smartphones last longer with code offload. Dissertation of Proc. of MobiSys, San Francisco, CA, USA, 2010.

[18] Young-Woo Kwon, Eli Tilevich. Energy-Efficient and Fault-Tolerant Distributed Mobile Execution. In 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), pp.18-21 June 2012

[19] Nikodin Ristanovic, Jean-Yves Le Boudec, Augustin Chaintreau, Vijay Erramilli. Energy Efficient Offloading of 3G Networks. MASS '11 Proceedings of the IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, pp.202-211, 2011.

[20]Yeli Geng, Wenjie Hu, Yi Yang, Wei Gao, Guohong Cao. Energy-Efficient Computation Offloading in Cellular Networks. IEEE 23rd International Conference on Network Protocols (ICNP), pp.145-155, 2015.