2019 电子科技大学美国数学建模竞赛模拟赛

承 诺 书

我们仔细阅读了数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们的题目编号是（填写：A 或者 B 或者 C）：B

我们报名队伍号是：H-213

参赛队员姓名学号(中文填写打印并签名): 1.田凯元

2.常开颜

3.陈瑞霖

指导教师或指导教师组负责人 (有的话填写)：

是否愿意参加 2019 年美国赛（是，否）：是

- 2017 级及 2016 级无国赛经历同学请先填写报名链接：https://www.wjx.cn/m/29408126.aspx
- 有国赛经历的队伍可以直接联系数模导师后登录我校数学建模网站进行报名

日期：2018 年 11 月 16 日

报名队号（请查阅《电子科技大学 2019 年美国赛模拟赛官方报名信息表+队号（按照队长姓名首字母 A-Z 排列）-最终版》后填写）

Machine Learning in Smart Thermostat

Summary

With the development of artificial intelligence, the smart home climate control system has been favored by people. It will automatically adjust the temperature of the room to meet people's comfort. This paper presents a series of feasible methods for smart home climate control systems.

In task one, we divide the task into two steps:

First, how to set the temperature to the user's comfort temperature? Based on the big data of a family in the laboratory of the University of Massachusetts, using **LSTM neural network method**, nine sets of environmental information were selected to construct a predictive temperature system.

Second, when to turn the thermostat on and off? Based on the schedule data of a certain user of Facebook, using **MMF (Position Prediction Algorithm for Multidimensional Mixed Features)** to construct a forecasting itinerary system. Forecast based on three rules. First, the periodicity of the user's schedule. Second, spatial clustering. Third, the user's preferences.

In the task two:

We compared our system with two other smart home climate control systems currently in use. One is **Ecobee**, another is **Xiaomi**. We chose three dimensions for comparison, which are **smart features**, **service** and **link**. Our products have achieved good results. And based on this analysis of our potential advantages and possible problems.

In the task three:

Based on the results of the first question, we continue to explore the situation of multiple thermostats for multiple people for a larger home.

The thermostat center temperature is predicted by task one. because the farther away from the center, the closer the temperature is to room temperature, the field is used to construct the temperature field according to Euclidean distance.

The room sensors can sense which rooms are occupied. Understanding which rooms are occupied helps the device prioritize, and it contributes to meet the needs of a larger home.

Key Word: LSTM MMF Thermostat

Contents

1 Introduction

1.1 Background

In recent years, Smart home devices are a hot industry. And everyone wants to be comfortable and cozy while at home. Today's smart temperature-controlled furniture can be remotely operated using a mobile app. However, users still need to set the temperature themselves.

It would be ideal if we could design a thermostat with machine learning capabilities. And this is exactly what the industry manufacturers are currently seeking.

1.2 Problem Statement and Analysis

Statement

1. To begin, assume that you live in a small home or apartment with only one thermostat (one heating/cooling zone).

a) Model a future smart home climate control system. Develop an algorithm or algorithms for this system to perform the functions described above. Discuss how your system works.

b) Discuss the potential benefits and possible issues with your system, and compare your system with at least two other smart home climate control systems currently in use.

2. Describe and discuss how you would make changes to your system for a larger home with additional thermostats to control several heating/cooling zones in the home.

3. Write a one-page non-technical *News Release* describing your new system.

Analysis

The core issue is how to synthesize the data sent by the sensor and the user on the central processor. The scheme for abstracting mathematical problems from intelligent temperature systems is detailed in Figure 1:

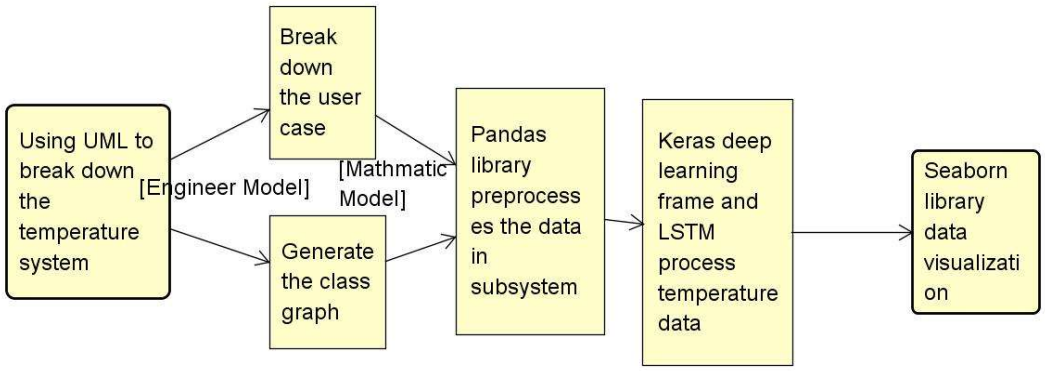


Figure 1: Engineering Modeling Process Flow

Object-Oriented Analysis - Use Case Modeling

The intelligent temperature control system must have a user-oriented graphical interface that can be explicitly programmed on the interface. At the same time, the system can also learn a model based on the user's data, and learn a fixed pattern according to the number of users and various data sent by the sensor.

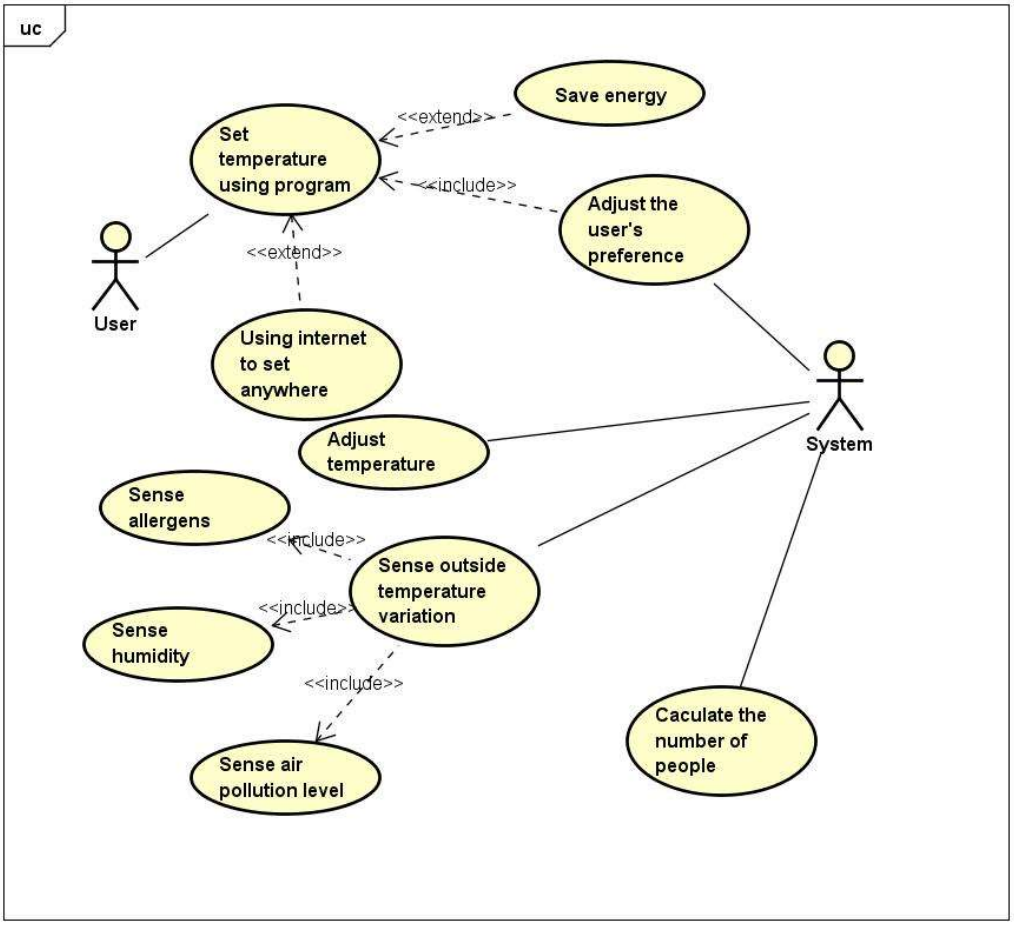


Figure 2: Temperature Control System Requirements Analysis Use Case

The architecture framework design - MVC framework

We designed an architecture framework for users and intelligent temperature systems based on MVC. The MVC framework is an architectural framework consisting of three parts: a model object, a view object, and a controller object. The controller object is instantiated into a temperature system central processor, the view object is a human-computer interaction window, and the model object specifically refers to the LSTM neural network model.

Mathematical model generation - class-based analysis

The following is based on scenario modeling to further analyze requirements and system design. Class diagrams map scenes into abstract objects and abstract mathematical models. The essence of the programmed temperature setter is the above described graphical temperature control interface in the MVC framework. The method in the programmed temperature object converts the user's input and programmed program into mobile data and sends it to the remote terminal. The remote terminal is

able to perceive the message. The remote terminal simultaneously sends data to the core processor in the MVC framework, in short, a server. The servers set up in the Internet of Things are embedded-oriented micro servers.

In summary, in the class-based modeling process, the actual problem of intelligent temperature system design is finally reduced to a mathematical problem—processing data sent by various sensors and users. In the following process, we will transform this modeling of engineering into modeling mathematical problems, and discuss the mathematical problems in the central controller in more detail.

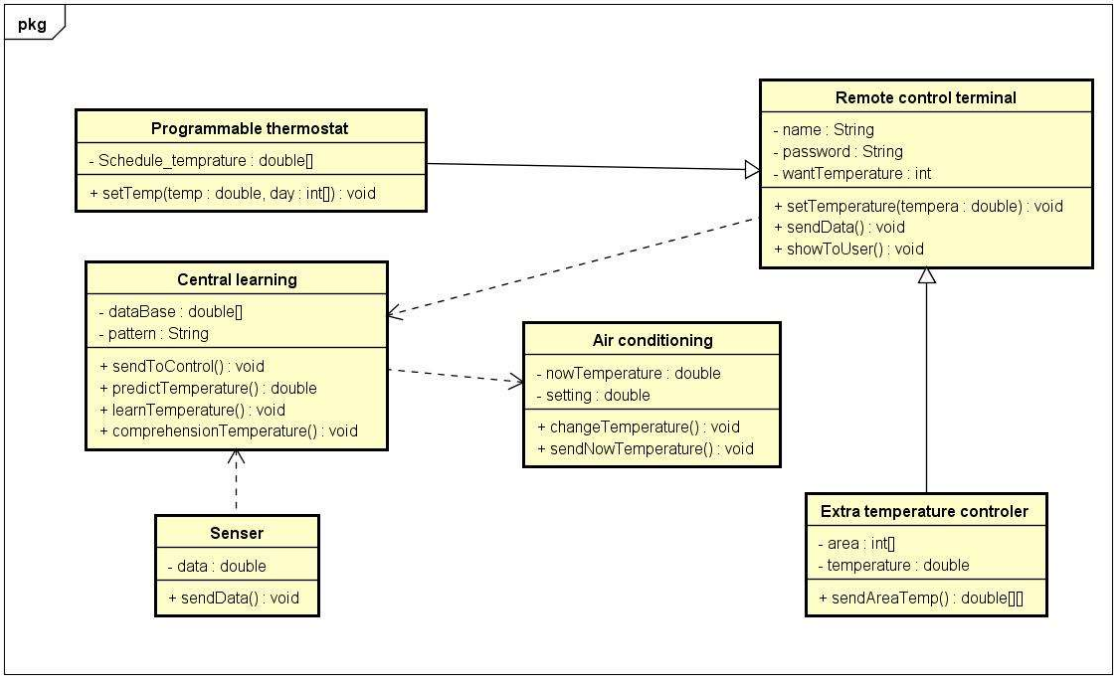


Figure 3: Class-based Modeling

2 Assumption and Symbol Explanation

2.1 Assumption

- Assume that the sensor is installed in the room.
- Assume that user's phone supports GPS.
- Assume that multiple rooms in the home are connected.
- Assume that the user wears clothes as normal thickness.

2.2 Symbol Explanation

Symbol	Definition
f_{norm}	Normalized feature
Δr	User's moving distance
β	Power-law distribution parameter

d_i	Distance the user travels once
f_{t1}	Time characteristic
t_{given}	Given prediction time
C_u	User's historical check-in data set
x_n'	Independent variable of the input data
t_n	True value corresponding to x_n'

3 Task 1

3.1 Data Pre-processing

3.1.1 Data Collection

This article uses deep learning methods to analyze data, so it needs to collect more data. The collected data is 8760, from the University of Massachusetts laboratory, reflecting the indoor and outdoor environmental changes, as well as the body's somatosensory temperature. The data set has a total of 9 attributes, respectively temperature, icon, humidity, apparent Temperature, pressure, windspeed, time, wind Bearing and dewpoint. Through the data set analysis, find out the interaction law between the data. Finally, Verify the LSTM neural network model.

Symbol	Interpreter
Temperature	The inside temperature
Icon	The weather condition
Humidity	The inside humidity
apparentTemperature	Sensible temperature
pressure	Atmospheric pressure
windspeed	Wind Speed
time	Local Time
windBearing	Wind direction
dewpoint	Dew point temperature

Table 1: Data Symbols

3.1.2 Data Pre-processing

Figure 4 shows the data processing required to be performed by building a data flow graph.

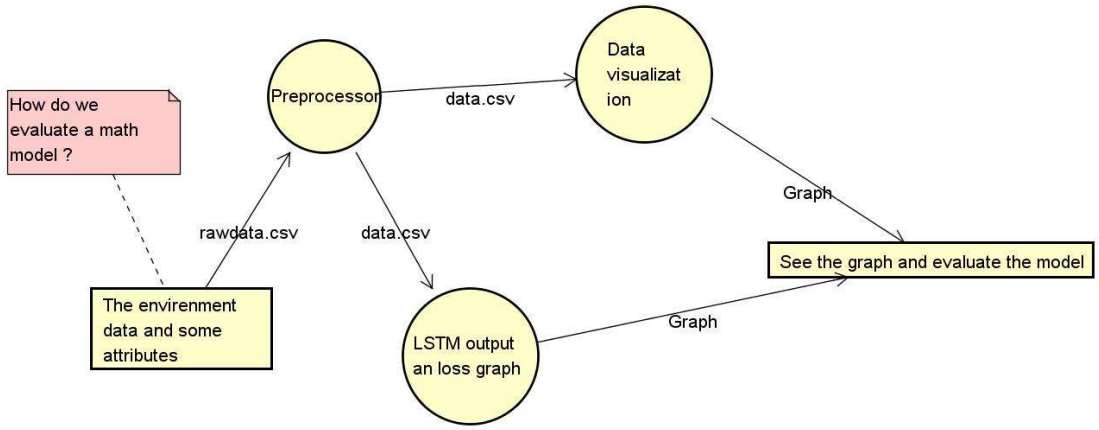


Figure 4: Data Preprocessing Process

Through the analysis of the data visualization image, find the general law of the data, then use LSTM neural network regression data. However, the data set has problems such as missing data, existence of string attributes and other partial data that are not standardized.

Solution of Missing Data

The missing data is concentrated in three columns, and the missing amount is not large, so we use the missing value filling method instead of the missing value culling method. According to scholars' research, continuous data is filled with the average of the attribute, while discrete attributes use the majority to fill the data. Since the missing data are all continuous data, we fill it by the average method.

User Data Filling Scheme

The original dataset did not provide a response from the user to the specific environment, so we tried to fill a small portion of the temperature as a simulated user, reflecting our own experience with the environment. After 80 manual judgments, we have mastered the regulation of temperature. And manually fill down on a large scale according to the law. After the feasibility analysis of the scheme, this artificial filling method simulates the actual behavior of the user and can better reflect the related operations of the user under the real system.

Coding Problem with Non-numeric Attributes

In the original data, a string such as {'clear-night', 'wind', 'partly-cloudy-night', 'rain', 'cloudy', 'fog', 'partly-cloudy-day', 'clear-day', 'snow'} appears in the icon field representing the outdoor weather condition. It needs to be mapped to an integer. Considering that the input data of the end user may be related to the intensity of light, we encode the light intensity according to different weather conditions from weak to strong in order to 0-9. Create numerical conditions for the next matrix calculation.

The Generation Scheme of The Time Series

Because the data set provides time fields, data preprocessing needs to be sorted according to the time field to reflect the characteristics of the time series. Since the LSTM neural network is a time series network, we need to create a sliding window

based on the original data. We set the length of the sliding window to be 2, and each window slides by one unit.

The Normalization of The Data

The basic principle of neural networks is to map linearly inseparable data into high-dimensional space and enhance the linear separability of data. Data normalization allows the data to be concentrated in a small range, which not only enhances the nonlinear mapping ability of the activation function, but also enhances the performance of the data set. The normalization process we use is as follows:

$$f_{norm} = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \tag{3.1}$$

3.1.3 Data Visualization Analysis

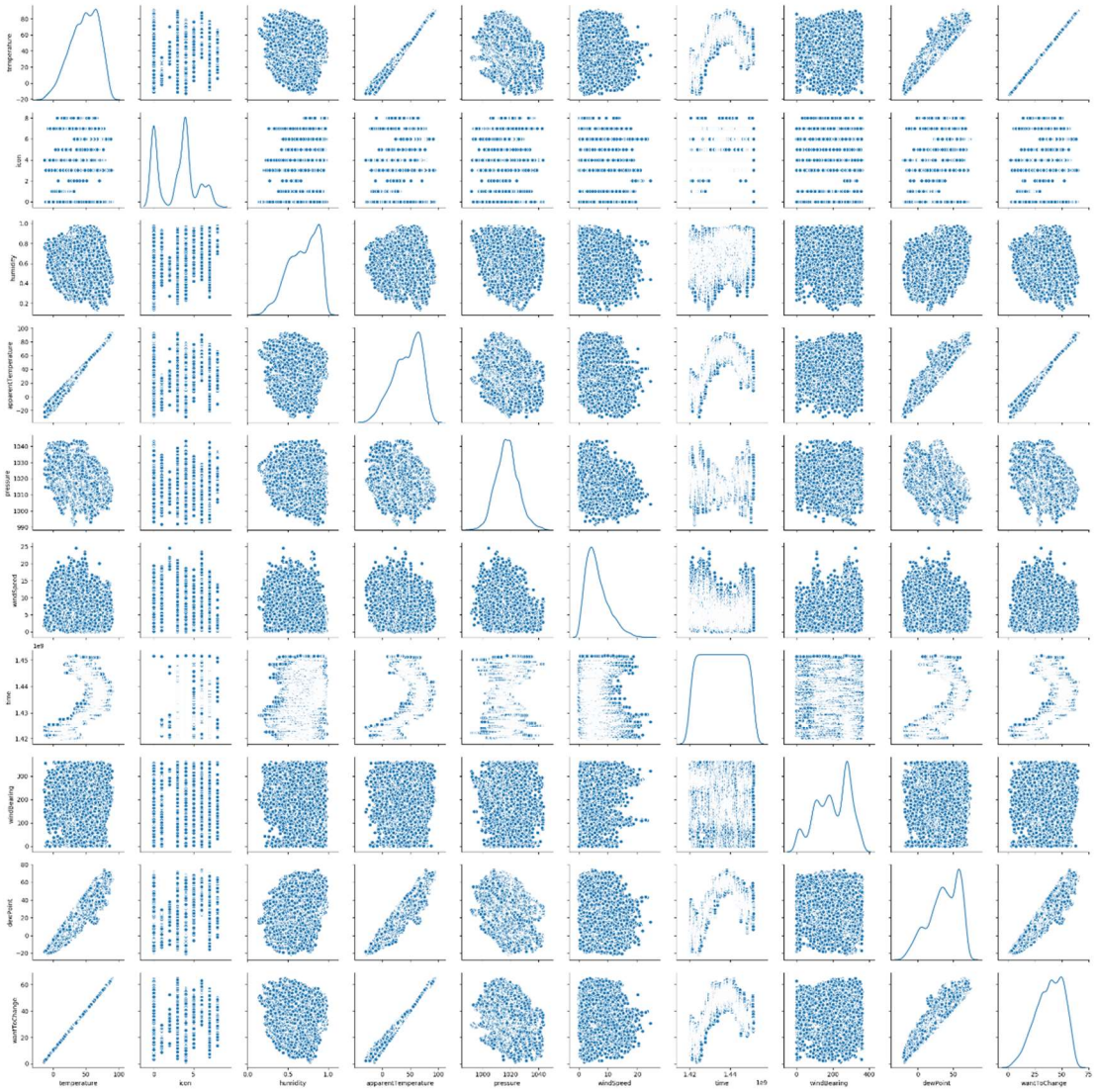


Figure 1: The scatter and distribution Graph

Figure 5 shows the distribution of the data set. The horizontal axis from left to right is temperature, icon, humidity, apparentTemperature, pressure, windspeed, time,

windBearing, dewpoint, wantToChange. The vertical axis is the same as the horizontal axis from top to bottom. The diagonal lines show the distribution of different attributes. It can be clearly found that the distribution of the data mostly approaches the normal distribution, which is also consistent with the results of the law of large numbers. The more the number of experiments, the closer the distribution is to the normal distribution.

The time attribute in the scatter plot has a fairly pronounced periodicity, which indicates that the temperature and environment related data and information are periodically changing over time.

3.2 LSTM and MMF model

3.2.1 LSTM method model

The standard LSTM neural network module consists of four interactive neural network layers, which avoids the problem of gradient disappearance or explosion when the number of iterations increases, and ensures that the model is fully trained, thus improving the prediction accuracy of the model.

$$\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)} \tag{3.2}$$

$$sigmiod(x) = \frac{1}{1+\exp(-x)} \tag{3.3}$$

Where σ is sigmoid and tanh is tanh function.

Figure 6 is our network structure based on the Keras framework design temperature set LSTM: 140200362874080 is the total number of input data elements, including attributes and iterations.

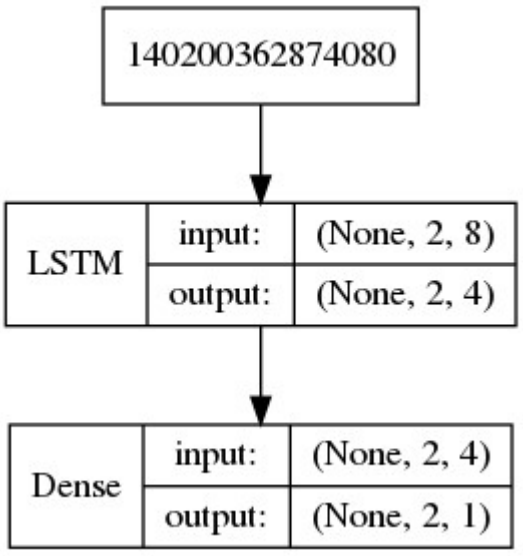


Figure 6: Network Structure

Figure 7 shows the loss reduction curve during the fitting of the LSTM model.

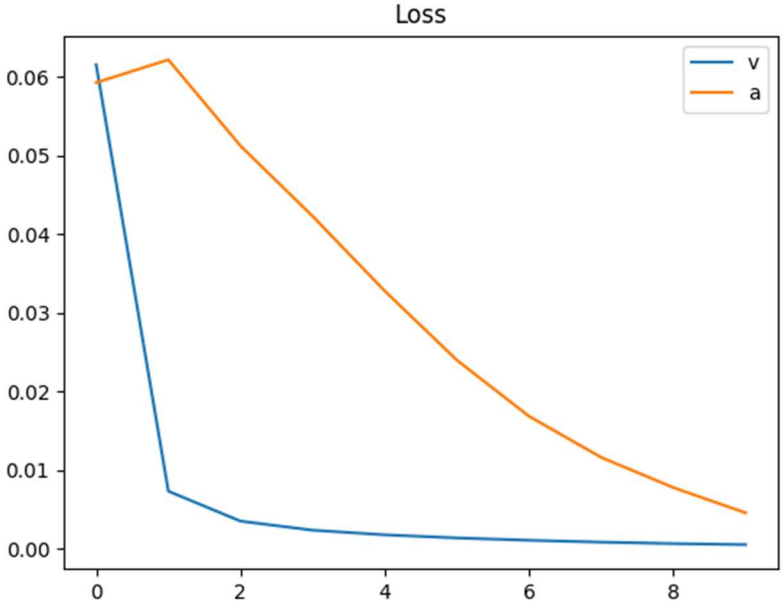


Figure 7: Loss Reduction Curve

V stands for validation set, *A* stands for train set. The ratio of the validation set we used to the sample population is 0.33. It can be seen that with the minibatch method, the drop loss is already low when the iteration is less than 10 times. We believe this is because the setting of the simulated user temperature is biased towards a linear setting. Therefore, the model can quickly complete the analysis of user data with fewer iterations.

Because we don't know what the actual temperature the user sets. Next, we use the Adagrad optimizer to find the actual temperature of the user design.

$$g_t = \nabla_\theta J(\theta_{t-1}) \tag{3.3}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha \cdot g_t}{\sqrt{\sum_{i=1}^{t} g_t^{\,2}}} \tag{3.4}$$

Let $g_{t,i}$ be the gradient of the i th parameter of the t-th round:

$$g_{t,i} = \nabla\Theta J(\Theta i) \tag{3.5}$$

Therefore, the process of parameter update in SGD can be written as:

$$\theta_{t+1,\ i} = \theta_{t,i} - \frac{\alpha \cdot g_{t,i}}{\sqrt{\mathrm{G}_{t,i} + \varepsilon}} \tag{3.6}$$

Where $G_t \in R^{d \times d}$ is a diagonal matrix, and each diagonal position *(i, i)* is the

sum of squares of the gradient of the corresponding parameter θ_i from the first round to the t round. ε is a smooth term and generally takes the value e^{-8}.

Using of LSTM

The above theoretical part has already stated that LSTM is theoretically feasible for temperature data analysis. This section shows that LSTM subsystems for Central learning can still provide the best experience for users.

User input data, input time, and sensor data are first recorded. Then wait for the amount of data entered by the user to reach a minibatch and then concentrate on the LSTM neural network. At this point, the user can access the Internet of Things Center through the mobile phone application, and program the temperature on the mobile phone to achieve the effect of temperature control. The control subsystem of the IoT Control Center responds at any time based on data sent by the sensor. Therefore, control and calculation separation can be achieved to minimize power consumption.

3.2.2 Analysis of Position Prediction Algorithm

This paper further analyzes the composition of the user's sign-in. Fig. 8 is a frequency histogram of each check-in location category in the Foursquare data set. Relatively speaking, the “Food” category and the “Shop & Service” account for the majority of the user's sign-in behavior, showing that users tend to share their locations when they are eating, shopping and travelling. The activities of eating out, shopping, etc. are often carried out in the evening, and the user's sign-in behavior during the activities explains the peak of the statistical frequency in the weekly sign-in mode.

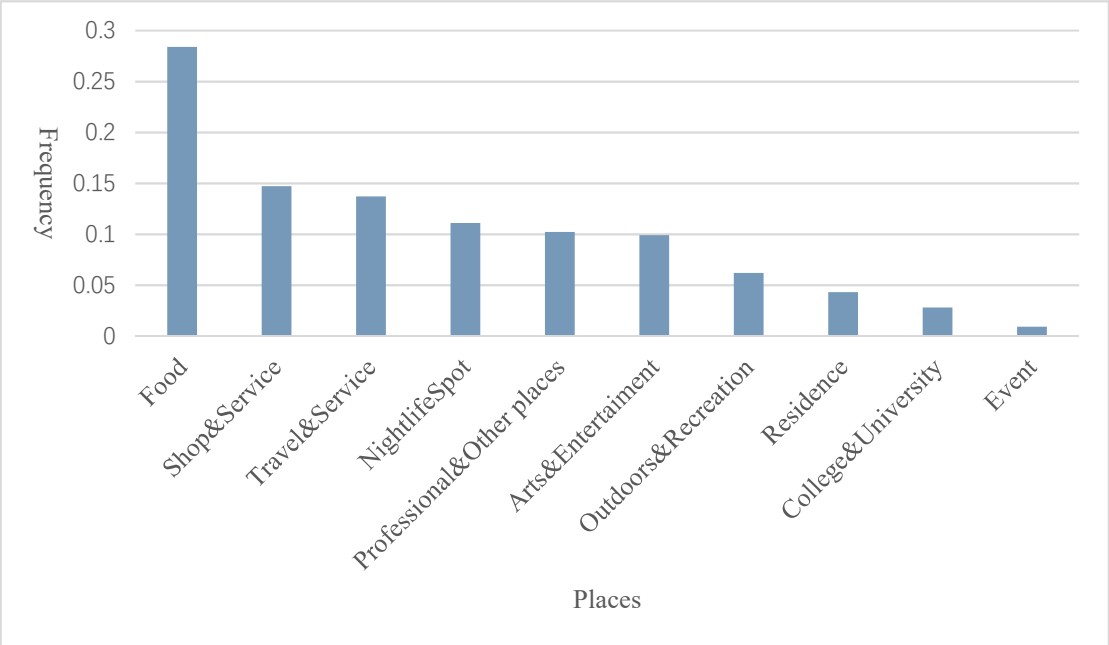


Figure 8: Sign in Places

Time Periodic Analysis

Numerous studies have shown that human activities have a strong time period. The related literature suggests that the weekly model is the two main periodic expressions

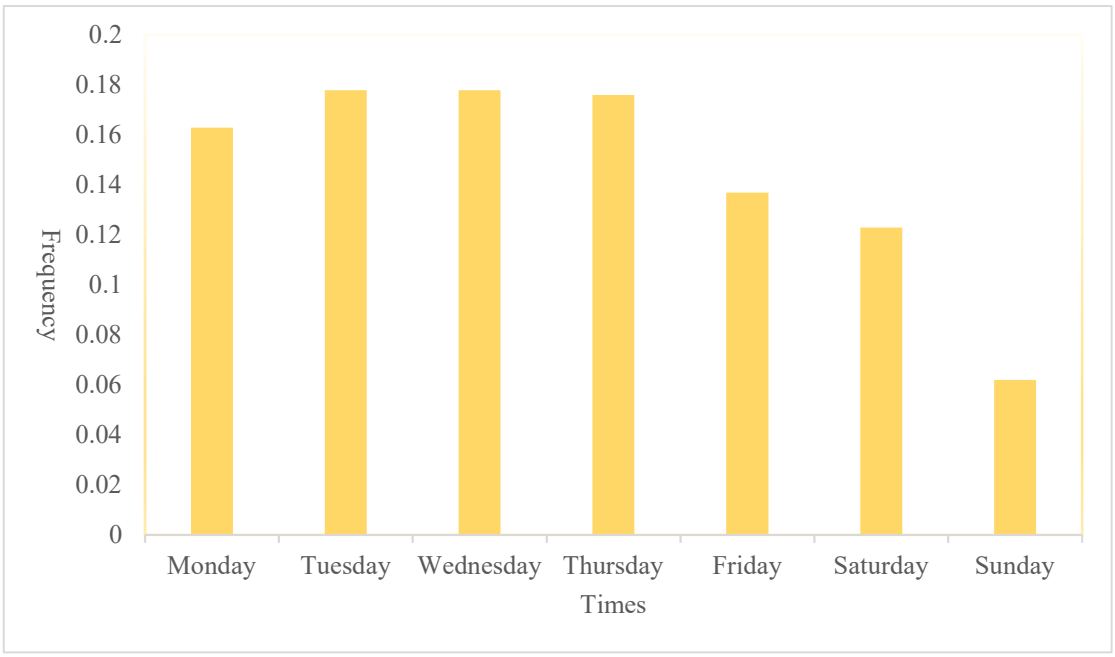
of sign-in behavior, that is, the week is the main cycle of human activities. In the analysis of this paper, the time period model will be used, and the check-in data will be statistically analyzed and analyzed in terms of hourly granularity, and the relationship between the sign-in location and time periodicity will be further discussed.

It can be seen that in the weekly check-in mode, the distribution frequency of the check-in frequencies at different times during the working day is relatively consistent, and the user has the highest frequency of sign-in on Thursday. There is a big difference in the frequency of sign-in on weekends and the workdays. This is because users often do different activities on weekends than on weekdays.

Since the user's different location categories represent a large number of location points, the users have different check-in time preferences for different categories of locations. In this paper, the three categories of "Arts & Entertainment", "Nightlife Spot", "College & University" are selected for user sign-in preference analysis.

Fig. 9 shows the distribution of check-in preferences for different location categories in weekly mode.

Users have a check-in under the "College & University" category during the week, but the frequency of sign-offs on Friday, Saturday and Sunday will be greatly reduced. On Sundays, the sign-in will be less than half of the normal working day, and "Nightlife Spot" will be presented. The rules are completely different. Users tend to do this activity on Friday, Saturday and Sunday. “Arts & Entertainment” and “Nightlife Spot” show similar distribution.



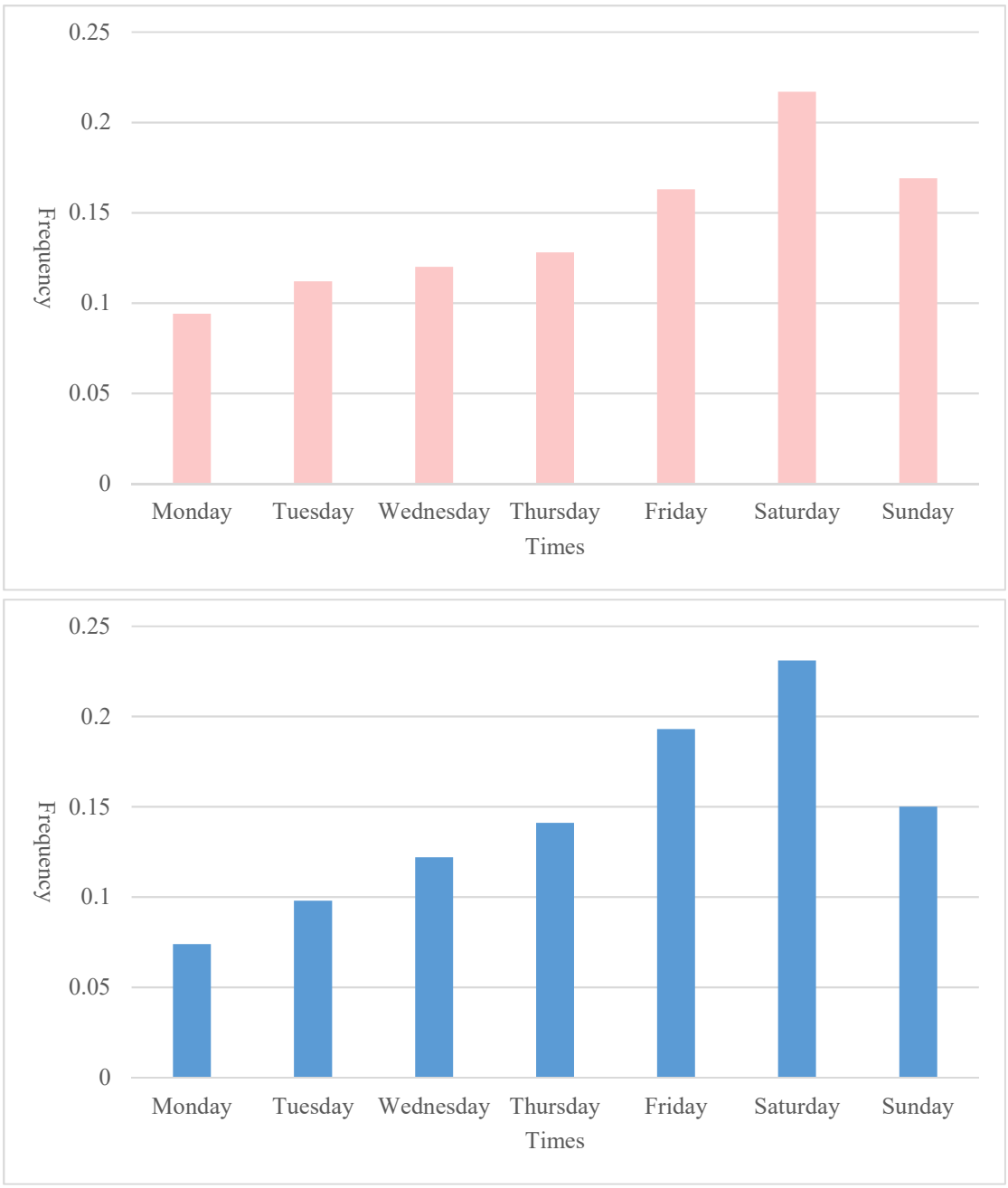


Figure 9: Three Places

Calculate all the single travel distances of New York users in the dataset and perform statistics, indicating that the distance of a single trip is exponentially distributed. This distribution largely reveals the clustering of the user's sign-in in the space: the user has an activity center, and the user's sign-in frequency is higher in the range closer to the sign-in center. The farther the user is from the sign-in center, the frequency of the user sign-in the lower. It can be concluded that the clustering of user research can be used to make user location predictions.

Spatial Factor Analysis

There are objective laws in human mobile behavior. Generally speaking, from a position, limited by human mobility and real geographical conditions, humans are more likely to move closer to the location, and the probability of movement decreases as the

distance increases.

This paper analyzes the obtained GPS trajectory data, and obtains the power distribution of the user's distance distribution satisfying the exponential truncation:

$$P(\Delta r) = (\Delta r + \Delta r_0)^{-\beta} \exp(-\frac{\Delta r}{k}) \tag{3.7}$$

Where Δr is the user's moving distance, and β is the power-law distribution parameter, and k is the truncation coefficient.

Inspired by this, this paper analyzes the obtained Foursquare New York user check-in dataset, and sorts the data in the check-in dataset C of a given user u by time. Get $S_u = \{< u, v_1, t_1 >, < u, v_2, t_2 >, < u, v_3, t_3 >\}$, user order adjacent to the distance between two check-ins is:

$$d_i = dis(v_i, v_{i-1}) C 7 \times 24 \tag{3.8}$$

d_i is the distance the user travels once. The distance $dis(v_a, v_b)$ between two positions v_a, v_b is calculated as follows:

$$dis(v_a, v_b) = 2R_{earth} \times \sin^{-1}(\sqrt{\phi}) \tag{3.9}$$

$$\phi = \sin^2[\frac{1}{2}(lat_a - lat_b)] + \cos(lat_a)\cos(lon_b)\sin^2[\frac{1}{2}(lon_a - lon_b)] \tag{3.10}$$

Where R_{earth} is the radius of the Earth and lon_a, lat_b, lon_b, lat_a is the longitude and latitude of the two places.

3.2.3 Position Prediction Algorithm for Multidimensional Mixed Features

Based on the analysis of time periodicity and spatial factors, the paper proposes a position prediction algorithm based on multidimensional mixed features.

Time-related Feature Extraction

From the periodic analysis of user behavior, the user behavior shows different distributions 24 hours a day, 7 days a week, so we divide the time into 7×24 time divisions, treat the absolute time as weekly mode, and the time granularity is hour.

The extraction time characteristic is: f_{t1}

For a single user, in the weekly time mode, the frequency at which the candidate position appears at a given time is quantized as:

$$f_{t1} = \frac{|\{c \mid (c \cdot v = v_{candidate}) \bigcap (r_w(c \cdot t) = r_w(t_{given})), c \in C_u\}|}{|\{c \mid r_w(c \cdot t) = r_w(t_{given}), c \in C_u\}|} \tag{3.11}$$

Where $v_{candidate}$ is home, t_{given} is the given prediction time, C_u is the user's historical check-in data set, and c is the check-in triplet consisting of the user, check-in time, and check-in location.

Extraction of spatially related features

This paper uses a piecewise function to describe the relationship between the user's check-in frequency and the location of the check-in and the home distance. The distance between the check-in location and the sign-in center is linear between 0~20km and 20~100km. Therefore, this paper uses a piecewise power-law distribution to describe the relationship between the user's check-in frequency at a given location and the distance between the location and the user's check-in center home. The expression of the power law distribution is as shown in the formula:

$$y = a \times x^b \tag{3.12}$$

Where a, b is the power-law distribution parameter, x is the distance from the given check-in location to the user's check-in center and home, and b is the user's check-in frequency at the given location.

By taking the logarithm of the equation, we can get the equation for linear fitting:

$$\log y = w_0 + w_1 \log x \tag{3.13}$$

Where $a = 2^{w_0}$, $b = w_1$. Further, let $y' = \log y$, $x' = \log x$, then the equation can be changed to:

$$y'(x', \vec{w}) = w_0 + w_1 \cdot x' \tag{3.14}$$

w_0 and w_1 are the coefficients of the equation and are uniformly represented by the vector $\vec{w}$. In this paper, the least squares method is used to fit the coefficients of the linear equation. To avoid over-fitting, a penalty factor is added to limit the coefficients:

$$E(\vec{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y'(x_n', \vec{w}) - t_n\}^2 + \frac{\lambda}{2}\left\|\vec{w}\right\|^2 \tag{3.15}$$

Where $E(\vec{w})$ is the loss function, N is the number of data input in the data set, x_n' is the independent variable of the input data, t_n is the true value corresponding to x_n', and λ is the regular term coefficient. The best fit model can be obtained when the loss function takes the minimum value. The minimum value can be obtained by finding the partial derivatives of w_0 and w_1 for $E(\vec{w})$ and then making them equal to *0*. In this case, w_0 and w_1 can be obtained, and the coefficients a and b of the power-law distribution can be obtained through conversion. In summary, the optimal a,b value is obtained by minimizing the loss function $E(\vec{w})$:

$$opt(a,b) = \arg\min_{a,b} E(\vec{w}) \tag{3.16}$$

According to the above process, the spatial feature f obtains the following quantitative formula:

$$f_s = \begin{cases} 0.5708 \times dis(l,h)^{-1.4979} & 0 < dis(l,h) < 20km \\ 6.2696 \times dis(l,h)^{-2.3833} & 20 < dis(l,h) < 100km \\ 0 & 100km \le dis(l,h) \end{cases} \tag{3.17}$$

According to the distance between the current location of the user and the home, the above formula can obtain the check-in frequency in this area as a feature value for predicting whether the user goes home.

The check-in frequency of each selected location and the check-in frequency of the location category at the candidate location are respectively extracted to obtain the user's sign-in preference feature $\{f_{p1}, f_{p2}\}$, which express the user's preference for the given location and the location category to which it belongs.

Due to the dimension of each feature and the range of its range, the calculated feature values need to be normalized, and the maximum and minimum normalization methods are used. Assuming that the quantized value of a feature is f, the quantized range of the feature is $[f_{\min}, f_{\max}]$, the feature is normalized as:

$$f_{norm} = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \tag{3.18}$$

Where f_{norm} is the normalized feature.

Position Prediction Algorithm Based on Multidimensional Mixed Features

The position prediction problem in this paper can be regarded as a classification problem, which is divided into two situations: check-in and no-sign-off.

First, according to the previous analysis, the quantized feature vector used in this paper is obtained as:

$$\vec{\alpha} = (Eig(f_1), Eig(f_2), Eig(f_3), Eig(f_4), Eig(f_5), Eig(f_6), Eig(f_s), Eig(f_f), Eig(f_{p_1}), Eig(f_{p_2}), 1)$$

The constant *1* is added using the logistic regression distribution formula. Based on the feature vector $\vec{\alpha}$, logistic regression can be used to classify whether the user is checked in at an alternate location for a given time.

According to the logical distribution formula, the probability that user *u* is checked in at position *v* at a given time is:

$$\rho = \frac{1}{1 + e^{-\vec{\alpha}\vec{\theta}}} \tag{3.19}$$

Where vector $\vec{\theta}$ is the corresponding weight of each feature in the probability prediction model in the eigenvalue question, which can be obtained based on the training set.

In summary, based on LPMMF (location prediction based on multidimensional mixture features), we can get this algorithm:

Input: $Feature = \{f_{t_1}, f_s, f_f, f_{p_1}, f_{p_2}\}$, t_{given}

Output: ρ

$L_c \leftarrow \text{initializeCandidatwSet}(u_0)$

Get $\vec{\alpha}$ according to t_{given}

Calculated probability: $\rho_i = \frac{e^{\alpha\theta}}{e^{\alpha\theta} + 1}$

Where L_c represents the set of candidate locations to be predicted by the user, and the first row initializes the candidate locations to be predicted, and selects the location that the user in the user dataset has visited as the candidate location. Line 2 calculates the feature vector based on the given time and user history check-in record and its social relationship. Line 3 calculates the sign-in probability of home based on the feature weight vector 0 obtained from unsupervised learning training. It can be seen from the algorithm that how to obtain the feature weight vector is one of the keys to guarantee the performance of the prediction algorithm. In this study, the maximum likelihood estimation method is used to estimate the model parameters based on the

training set, and the feature weight vector $\vec{\theta}$ is obtained.

4 Task 2

4.1 Product comparison

Ecobee:

Ecobee is an intelligent thermostat that Apple introduced in 2015 and currently ta kes a large market share, so we chose this product for comparison.

XiaoMi:

XiaoMi air conditioning is xiaomi company in 2016 launch of a smart air conditi oning, but there is no industrial chain.

4.1.1 Smart Features

SMART FEATURES	Our product	Ecobee	XiaoMi
Programmed Scheduling	Automatic prediction	Manual	no
Follow user	yes	yes	no
Intelligent temperature	yes	yes	yes
Sunblock Feature	yes	yes	no
Ambient Light Sensor	yes	no	no
Different Room Different Temperature	yes	no	no
Energy Savings (avg. determined by manuf)	23%	10-12% on heating; 15% on cooling	15%

Table 2: Smart Features

Follow user: Follow user is an occupancy detection technology that is able to detect exactly which room you are in and adjust settings and temperature accordingly.

Sunblock Feature: Our product has a pretty nifty ‘sunblock’ feature, which detects when the thermostat is in direct sunlight, being aware that the temperature reading will be higher than it actually is.

Different room different temperature: Our product room sensors also sense which rooms are occupied. Understanding which rooms are occupied helps the device prioritize, and it contributes to its geofencing readings.

4.1.2 Service

Service	Our product	Ecobee	XiaoMi
Voice service	yes	yes	no
Compatibility	widely compatibility	lack the depth of features	less
More thermostats	net	individual	individual

Table 3: Service

Compatibility: Our products compatible with conventional (2H/2C), heat pump (H4/2C) including two-stage auxiliary heat, gas, oil, electric, or dual fuel systems.

This model also supports added accessories, like humidifiers, dehumidifiers, ventilator, ERVs, or HRVs. The Ecobee4 can connect to smart home devices like Amazon Alexa, Google Assistant, IFTTT, Apple Home Kit, Samsung SmartThings, or more using a 2.4 GHz b/g/n system.

4.1.3 Link

Link	Our product	Ecobee	Xiaomi
Network outage	infrared	no	no
Wi-Fi Issues	Do not have	have	Do not have
multistage systems	Internet of things	no	no

Table 4: Link

Wi-Fi Issues: The Ecobee is hardwired into the home and comes with a Power Extension Kit, for home which does not have a C-Wire, this means that the Wi-Fi connectivity is highly unlikely to ever drop out.

4.2 Advantages and Disadvantages

4.2.1 Advantages

Intelligent Temperature

The thermostat is a self-programmable and customizable thermostat. The thermostat is able to learn your temperature and habits preferences and can even learn when you are away and when you are present at home. Once you have the device setup, it instantly starts to learn based on your preferred temperature. So, after some time of installation, you won't need to make any adjustments yourself.

Predict Your Schedule

It learns more about adjusting the temperature of your house based on your activities and schedule like when you leave home, when you come back and when you go to sleep.

And if you don't go past its light sensors and 150-degree motion for some time, it will assume you are not home and change to the energy efficient away mode. If the thermostat finds you gone during that time period for a few days continuously, then it will learn this new pattern and then adjust the schedule accordingly.

Zoning Systems

If you have a bigger than average house and your HVAC is unable to maintain a consistent temperature in each room, then you need to invest in a smart thermostat with a zoning system.

Zoning systems usually come with a separate thermostat for each zone. This way, you can adjust the temperature of each zone accordingly. However, there are some thermostats that can streamline the whole HVAC system. Such smart thermostats can manage up to eight zones at the same time.

4.2.2 Disadvantages

Forecast Error

If the user's schedule very irregular, leading to predict the effect are not allowed, back to consume more energy.

Sensor Installation Problem

If sensor installation location bad, such as direct sunlight or ventilation bad, will be very affect system for the surrounding environment judgment.

Security

There is hidden danger that the itinerary is stolen. This is a common problem with the internet of things.

5 Task 3

5.1 Analysis of Distributed Temperature Controller Model

In task 1, we studied the model of single-center temperature controller using LSTM method and how to adjust indoor temperature under a temperature controller. We will study how to build a temperature distribution model with multiple controllers.

Here you might as well borrow the model of task 1. The central controller is defined first. The central controller is the temperature controller in the room where the predecessors were. This controller generates data even if we compute data from the LSTM model in the first question. The rest of the temperature controller is defined as an edge controller. In the case of large numbers of people and distributed in different rooms, multiple central controllers need to be set. There are several different edge controllers at the same time. This case is defined as a multicenter temperature controller model.

The temperature calculation of edge controller is studied by the model. Through experiments, we think that the temperature of the edge controller is subject to a certain distribution. With the increase of the distance of the main controller, the setting temperature of the edge controller is closer to room temperature. Unlike the original assumption of our experiment, we initially thought that the temperature drop model could be modeled by the KNN algorithm. With the development of the experiment, we find that KNN mainly focuses on discrete data clustering, the temperature controller can fit the continuous property of temperature by setting the temperature drop speed. Therefore, KNN is not suitable for this model compared with continuous decline. Similar to the mean square error loss defined in the first question, we still use the Euclidean distance to define the distance from the central controller, and use the continuous method to calculate the setting temperature of the edge controller.

5.2 Implementation of the model

Considering the existence of multiple central temperature controllers, we think that in this case, the nearest central temperature controller for any edge controller is the European distance that it wants to calculate. That is, for each edge controller program, just calculate its distance from the nearest central controller.

We will experiment on the distribution problem of the multi-center controller and give the specific formula and the result of the experiment. Similar to the object-oriented analysis described in the system analysis phase, the object-oriented approach is still used in this phase.

Programming language select object-oriented language java. Java language has good object-oriented characteristics, can better outline in the reality of the relationship between different classes, as well as their internal composition structure. Here we study the problem of the distribution of temperature set by the redundant controller center and the whole house temperature controller.

We simulated a large room model with 37 controllers, and in this big room we set up two central controllers, one in each central controller's room. The model of the central controller is stored in an array of characters, and we use the C-language model to visualize it. See appendix IV for details. In the following figure, the red represents the position of the central controller, while the blue represents the position of the edge controller. The upper left corner coordinates are (0,0), and the figure is represented by an array of room characters at 1:30.

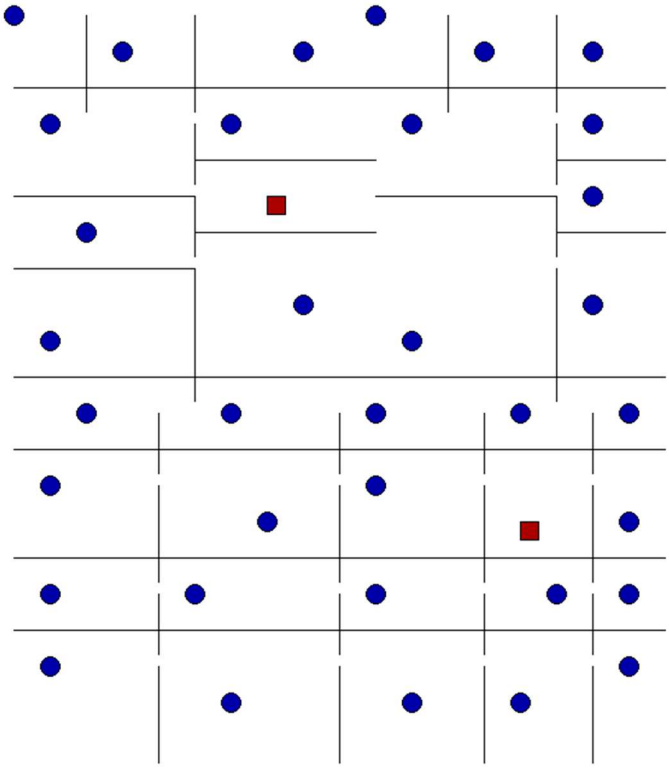


Figure 10: A Probable Room Model

Java code in the specific algorithm, using s to represent the edge of the controller's position with p representative and central controller's position. In the Java language code, the corresponding comments and the variable function description of word interpretation are given. The data of these algorithms are listed in appendix 3. In the algorithm, we set the decline rate variable to represent the temperature of the controller relative to the intermediate controller in each room approaching the room temperature rate. The larger the decline rate variable, the closer the controller in each room to the setting temperature of the central controller.

According to the distance of each room, with the distance of the central controller increasing, and the temperature approaching room temperature, we set a set

of functions that are negatively related to distance. To describe the setting temperature of the controller at different seasons.

The algorithm is as follows:

***distance**= min(sqrt((x-person_x)*(x-person_x)+(y-person_y)*(y-person_y)), sqrt((x-person2_x)*(x-person2_x)+(y-person2_y)*(y-person2_y)))*

if in winter then

setTemperature=min(centralSetTemperature,roomTemperature+(centralSetTemperature-roomTemperature)(1.0/(1+distance*distance))*declineRate)*

else if in summer then

setTemperature=max(centralSetTemperature,roomTemperature+(centralSetTemperature-roomTemperature)(1.0/(1+distance*distance))*declineRate)*

The engineering principles of our modeling are described below. The Internet of things is a typical embedded system, which requires high real-time performance, and can quickly calculate the result and control the entity under a limited resource environment. In the formula we use as many as possible finite four operations, we can get the edge temperature relatively accurate results.

The model is simplified, mainly reflected in the distance between the edge sensor and the multi-center sensor, and the selection depends only on the center sensor nearest to it. The central sensor farther away, though connected to him, is relatively weak. From the point of view of enhancing the reliability of embedded system, we ignore this connection, so this simplification is reasonable. From the point of view of system design, the second step of embedded system design needs to divide the interface between hardware and software. Here we make the software design part of the corresponding simplified processing, making the model more simplified. The cost of hardware has also fallen.

In particular, this is different from the first task, where we use the CPU as a server to process data. The CPU is fast, so LSTM large-scale neural network calculation can be realized. In the third task, however, the temperature of the air conditioner in the room in which it is located should be calculated on the basis of the data of each sensor. It is based on embedded microcontroller, which is not suitable for large scale operation. This problem belongs to the problem of edge calculation. For edge computing, we cannot use large-scale neural networks or complex formulas, otherwise the reliability and usability of the system will be greatly reduced. This is the biggest difference between this question and the first task, and the starting point of our algorithm design and model construction.

5.3 Model testing and validation

In the test model, we set the central controller and calculated the room temperature according to the LSTM model at 26 degrees centigrade, at 34 degrees centigrade. According to the coordinates of different controllers in each room, we give the corresponding calculated distance. The test results are shown in appendix vi.

The results of the test were about 37. Appendix V shows only one person at coordinates (7,5). Appendix vi shows the situation that two persons at coordinates (7,5) and (14,14) respectively. During the test, the user can input the decline rate variable to control the increase of the distance between the central controller and the room, and make the temperature drop or increase fast. Similar to the first question, we think that the rate of change can enable users to set up programming on their mobile phones, or as a parameter input system for LSTM networks, the corresponding decline rate variable is automatically calculated, but only if the user must provide enough learning data to support the network.

As a simulated user, we think this set of output data better reflects the desired temperature setting results. The setting model of room temperature under the multi-sensor and multi-control is realized in the task.

6 Task 4

Intelligent thermostat

With the rise of the “Smart Home”, consumers are beginning to realize all the wa ys a smart device can be used, but have you ever considered a smart thermostat?

So, we launched the smart home climate control systems. Once the smart thermostat is connected to your home Wi-Fi network, you can simply use your smartphone to log in and view the status of the thermostat regardless of where you are.

Learning thermostats

Our product calls its flagship thermostat a “learning” thermostat. It can learn from your preferences and adjust your home’s temperature to your liking automatically. After a few days or weeks of observing your habits and preferences, our product gathers enough information to understand when it should make automatic adjustments for you.

For example, if you like the temperature a few degrees lower while you sleep and you lower the temperature down to 65 degrees around the same time every night, eventually our product will understand this habit and make the adjustment for you.

With this feature, the device can understand your HVAC needs and then adjust the temperature accordingly. So, not only will the smart setback feature keep everyone more comfortable, but it will also help to reduce energy costs a significant amount.

And our products will predict when you will go home and open the thermostat in advance to get the comfortable temperature when you get home.

Zoning Systems

If you have a bigger than average house and your HVAC is unable to maintain a consistent temperature in each room, then you need to invest in a smart thermostat with a zoning system.

Zoning systems usually come with a separate thermostat for each zone. This way, you can adjust the temperature of each zone accordingly. However, there are some thermostats that can streamline the whole HVAC system.

Let’s say you and your family are in the basement, which tends to run colder than the rest of the house. If you’ve put a room sensor in the basement, our product will know that’s the room to prioritize. It can make sure you’re comfortable there and, while you and your family are in the basement, it will care a bit less about other rooms in the house.

We believe our product is the best smart thermostat, due to the incredibly intelligent nature of it. Our products have broad prospects in the future. Hope you like our products.

7 Strengths and Weaknesses

7.1 Strengths

- The formation of our models is simple, where the relationship between the factors and resultsis clear and straightforward. However, by introducing some, inclusive parameters andcomponents, the models are easily applied to the case of other smart homes. Thus, our models are also relatively universal.
- We take LSTM, Multidimensional Mixed Features, KNN algorithm. So, our model is convincing and substantial.

7.2 Weaknesses

- Although we have tried our best. Time is finite, and some data are missed. As a result, the missing data can still bring the errors in evaluation.
- In our model, large quantities of statistics are required to ensure the prediction is accurate and reliable. Thus, the modified cube model is dependent on the database to guarantee the accuracy.

8 References

[1] Ecobee3 Wikipedia
https://en.wikipedia.org/wiki/Ecobee
[2] UMassTraceRepository
http://traces.cs.umass.edu/index.php/Smart/Smart
[3] 人体舒适气候适应模型研究 李俊鸽 2006.11
[4] LSTM（Long Short-Term Memory）baidu
https://baike.baidu.com/item/LSTM/17541102?fr=aladdin

9 Appendix

Disclaimer: Please refer to my GitHub for source code. Due to page limit, we only show the core code. The relevant data set is also available on my GitHub.
https://github.com/changkaiyan/mathmodel

Appendix 1: Python code on Data preprocessing (Based on Pandas library)

```
#------------------------------------------------------
# Copyright 2018 Kaiyan Chang, Kaiyuan Tian, Ruilin Chen
# For the data preprocessing
# Python 3.6 for Linux. Only run in Linux system.
#------------------------------------------------------
#Read data from csv
raw_data=pd.read_csv("datar.csv")
temp=set(raw_data['icon'])
print(temp)

#Choose a best index for icon
size_mapping={label:index for index, label in enumerate(temp)}
raw_data['icon']=raw_data['icon'].map(size_mapping)

#Caculate the user defined temperature
sigma=0.2
mu=0
wanttochange=0.607*raw_data['temperature'].values+10.092+np.random.normal(m
u,sigma,raw_data.shape[0])

#Fill NaN
raw_data['wantToChange']=wanttochange
raw_data[raw_data.isnull().values==True]
raw_data['pressure']=raw_data['pressure'].fillna(raw_data['pressure'].mean())
raw_data['windBearing']=raw_data['windBearing'].fillna(raw_data['windBearing'].
mean())
```

```
raw_data['windSpeed']=raw_data['windSpeed'].fillna(raw_data['windSpeed'].mean(
))

#Sort by time
raw_data.sort_values(by="time")
raw_data.to_csv('datanoraw.csv',index=False)
sns.pairplot(raw_data,diag_kind='kde')
plt.savefig('datasetprocessingpairplot1')
raw_data
```

Appendix 2: Python code on Data regression (Based on Keras and tensorflow library)

```
#----------------------------------------------------------
# Copyright 2018 Kaiyan Chang, Kaiyuan Tian, Ruilin Chen
# For the data regression
# Python 3.6 for Linux. Only run in Linux system.
#----------------------------------------------------------

look_back=2    # A length of time window
epochs=1000     # training Epochs
batch_size=20     #In minibatch the batch size is 20


#Create a time windows and extends matrix to tensor
def create_dataset(dataset):
      dataX, dataY = [],[]
      global look_back
      for i in range(len(dataset) - look_back - 1):
            x= dataset[i:i+look_back, :dataset.shape[1]-1]
            dataX.append(x)
            y = dataset[i:i+look_back, dataset.shape[1]-1:dataset.shape[1]]
            dataY.append(y)
      dataX=np.reshape(np.array(dataX),((len(dataset) - look_back -
1),look_back,dataset.shape[1]-1))
      dataY = np.reshape(np.array(dataY), ((len(dataset) - look_back -
1),look_back,1))
      return dataX, dataY

#Define batch normalization class
scaler_x = MinMaxScaler()
scaler_y=MinMaxScaler()
scaler=MinMaxScaler()
```

```
#Read data from .csv
data_set = read_csv('datanoraw.csv', header=0, index_col=0)
data_set=data_set.values.astype('float32')

#Batch normalization
Train_x = scaler_x.fit_transform(data_set[:,:data_set.shape[1]-1])
Train_y = scaler_y.fit_transform(data_set[:,data_set.shape[1]-1:data_set.shape[1]])
data_set=np.hstack((Train_x,Train_y))
train = data_set[0:, :]
X_train, y_train = create_dataset(train)

#Define model using keras
model = Sequential()
model.add(LSTM(units=4, input_shape=(look_back, data_set.shape[1]-
1),return_sequences=True))
model.add(Dense(units=1))
model.compile(loss='mean_squared_error', optimizer='adam')

#Run the model
history=model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
verbose=2,validation_split=0.33)

#Model visulization
model.summary()
plt.plot(history.history['loss'], label='train')
plt.legend('train')
plt.plot(history.history['val_loss'], label='test')
plt.legend('validation')
plt.title('Loss')
plt.show()
plt.savefig('loss 1-10')
```

Appendix 3: Java Code for room model

```
/**
 * @file myRoom.java
 * @author changkaiyan (changkaiyan@std.uestc.edu.cn)
 * @brief Multi-central senser caculation
 * @copyright Copyright 2018 Kaiyan Chang, Kaiyuan Tian, Ruilin Chen
 *
 */
public class myRoom{
    //Person is in (7,5) (14,14)
    private char[][] room;//The data is too heavy you can see my github
```

```
	private Senser[] roomsenser=new Senser[64];
	private int sensercount=0;
	public static void main(String[] args){
		myRoom mr=new myRoom();mr.setSenser();
		mr.getRoomTemperature();

	}
	public void setSenser()
	{
		for(int i=0;i<room.length;++i)
		{
			for(int j=0;j<room[i].length;++j)
			{
				if(room[i][j]=='s')
				{
					roomsenser[sensercount]=new Senser(i,j);
					sensercount++;
				}
			}
		}
	}
	public void getRoomTemperature()
	{
		for(int i=0;i<sensercount;++i)
			roomsenser[i].printData();
	}
}
//I assume the room temperature is same in all rooms
class Senser{
	private int x;
	private int y;
	private double distance;
	private double setTemperature;
	//Multi-central-Senser
	Senser(int x,int y)
	{
		int person_x=7; int person_y=5;
		int person2_x=14; int person2_y=14;
		double centralSetTemperature = 26.0;
		double roomTemperature=35;
		this.x=x; this.y=y;
		double declineRate=35;//The recomment data
		this.distance=Math.min(Math.sqrt((x-person_x)*(x-person_x)+(y-
person_y)*(y-person_y)),
```

```
                    Math.sqrt((x-person2_x)*(x-person2_x)+(y-person2_y)*(y-
person2_y)));
            if(roomTemperature<centralSetTemperature)//Often Winter

setTemperature=Math.min(centralSetTemperature,roomTemperature+(centralSet
Temperature-
roomTemperature)*(1.0/(1+this.distance*this.distance))*declineRate);
            else//Often Summer

setTemperature=Math.max(centralSetTemperature,roomTemperature+(centralSet
Temperature-
roomTemperature)*(1.0/(1+this.distance*this.distance))*declineRate);
        }
        public void printData()
        {
            System.out.println("The senser location : ("+x+","+y+")    Distance is
"+distance+" The auto setting temperature is:"+setTemperature);
        }
}
```

Appendix 4: C/C++ code for room model viusalization

```
/**
 * @file C_code_model.cpp
 * @author changkaiyan (changkaiyan@std.uestc.edu.cn)
 * @brief Multi-central senser Model Visualization
 * @copyright Copyright 2018 Kaiyan Chang, Kaiyuan Tian, Ruilin Chen
 *
 */
int main()
{
   char room[18][20] ;//The data is too heavy you can see my github
   initgraph(1024, 768);setorigin(100, 100);
   setbkcolor(WHITE);cleardevice();setcolor(BLACK);
   for(int i=0;i<18;++i)
      for (int j = 0; j < 20; ++j)
      {
         if (room[i][j] == 's')
         {
            setfillcolor(BLUE);
            fillcircle(i * 30, j * 30, 8);
         }
         else if (room[i][j] == 'p')
         {
```

```
            setfillcolor(RED);
            fillrectangle(i * 30, j * 30, i * 30 + 15, j * 30 + 15);
        }
        else if (room[i][j] == '|')
        {
            setlinecolor(BLACK);
            line(i * 30, j * 30, i * 30+30, j * 30 );
        }
        else if (room[i][j] == '-')
        {
            setlinecolor(BLACK);
            line(i * 30, j * 30, i * 30, j * 30+50 );
        }
    }
  system("Pause");closegraph();
}
```

Appendix 5: Room model only one person(Run in Java 10)

```
Loc : (0,0)  Distance 8.602325267042627 Setting temper:30.80
Loc : (1,3)  Distance 6.324555320336759 Setting temper:27.32
Loc : (1,9)  Distance 7.211102550927978 Setting temper:29.06
Loc : (1,13)  Distance 10.0 Setting temper:31.88
Loc : (1,16)  Distance 12.529964086141668 Setting temper:33.01
Loc : (1,18)  Distance 14.317821063276353 Setting temper:33.47
Loc : (2,6)  Distance 5.0990195135927845 Setting temper:26.00
Loc : (2,11)  Distance 7.810249675906654 Setting temper:29.92
Loc : (3,1)  Distance 5.656854249492381 Setting temper:26.00
Loc : (5,16)  Distance 11.180339887498949 Setting temper:32.50
Loc : (6,3)  Distance 2.23606797749979 Setting temper:26.00
Loc : (6,11)  Distance 6.082762530298219 Setting temper:26.71
Loc : (6,19)  Distance 14.035668847618199 Setting temper:33.41
Loc : (7,14)  Distance 9.0 Setting temper:31.16
Loc : (8,1)  Distance 4.123105625617661 Setting temper:26.00
Loc : (8,8)  Distance 3.1622776601683795 Setting temper:26.00
Loc : (10,0)  Distance 5.830951894845301 Setting temper:26.00
Loc : (10,11)  Distance 6.708203932499369 Setting temper:28.15
Loc : (10,13)  Distance 8.54400374531753 Setting temper:30.74
Loc : (10,16)  Distance 11.40175425099138 Setting temper:32.60
Loc : (11,3)  Distance 4.47213595499958 Setting temper:26.00
Loc : (11,9)  Distance 5.656854249492381 Setting temper:26.00
Loc : (11,19)  Distance 14.560219778561036 Setting temper:33.52
Loc : (13,1)  Distance 7.211102550927978 Setting temper:29.06
Loc : (14,11)  Distance 9.219544457292887 Setting temper:31.34
Loc : (14,19)  Distance 15.652475842498529 Setting temper:33.72
```

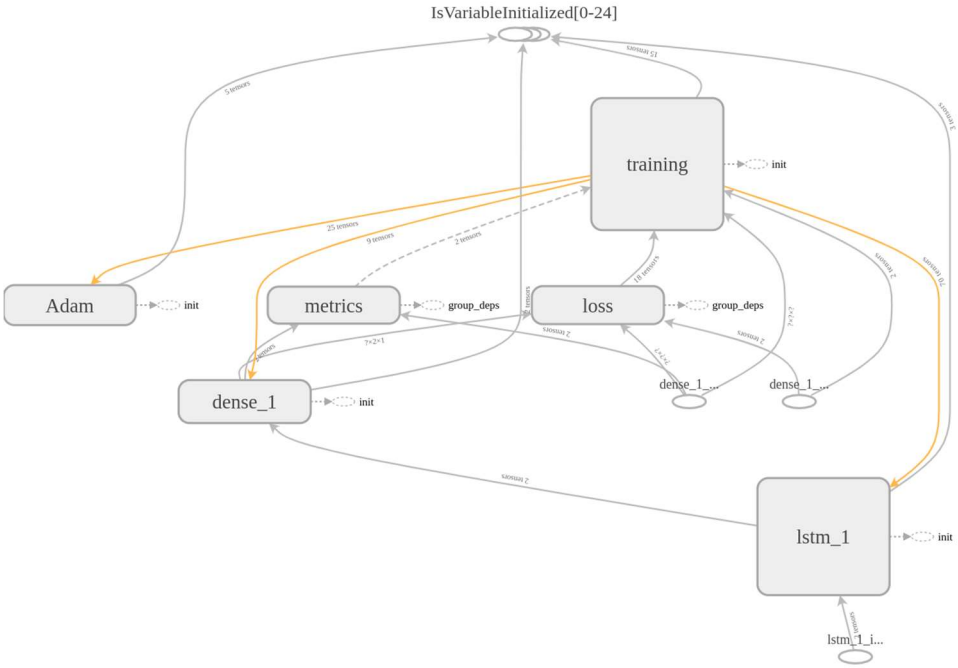
Loc : (15,16) Distance 13.601470508735444 Setting temper:33.31
Loc : (16,1) Distance 9.848857801796104 Setting temper:31.79
Loc : (16,3) Distance 9.219544457292887 Setting temper:31.34
Loc : (16,5) Distance 9.0 Setting temper:31.16
Loc : (16,8) Distance 9.486832980505138 Setting temper:31.54
Loc : (17,11) Distance 11.661903789690601 Setting temper:32.70
Loc : (17,14) Distance 13.45362404707371 Setting temper:33.27
Loc : (17,16) Distance 14.866068747318506 Setting temper:33.58
Loc : (17,18) Distance 16.401219466856727 Setting temper:33.83

Appendix 6: Room model for two persons(Run in Java 10)

Loc : (0,0) Distance 8.602325267042627 Setting temper:30.80
Loc : (1,3) Distance 6.324555320336759 Setting temper:27.32
Loc : (1,9) Distance 7.211102550927978 Setting temper:29.06
Loc : (1,13) Distance 10.0 Setting temper:31.88
Loc : (1,16) Distance 12.529964086141668 Setting temper:33.01
Loc : (1,18) Distance 13.601470508735444 Setting temper:33.31
Loc : (2,6) Distance 5.0990195135927845 Setting temper:26.00
Loc : (2,11) Distance 7.810249675906654 Setting temper:29.92
Loc : (3,1) Distance 5.656854249492381 Setting temper:26.00
Loc : (5,16) Distance 9.219544457292887 Setting temper:31.34
Loc : (6,3) Distance 2.23606797749979 Setting temper:26.00
Loc : (6,11) Distance 6.082762530298219 Setting temper:26.71
Loc : (6,19) Distance 9.433981132056603 Setting temper:31.50
Loc : (7,14) Distance 7.0 Setting temper:28.70
Loc : (8,1) Distance 4.123105625617661 Setting temper:26.00
Loc : (8,8) Distance 3.1622776601683795 Setting temper:26.00
Loc : (10,0) Distance 5.830951894845301 Setting temper:26.00
Loc : (10,11) Distance 5.0 Setting temper:26.00
Loc : (10,13) Distance 4.123105625617661 Setting temper:26.00
Loc : (10,16) Distance 4.47213595499958 Setting temper:26.00
Loc : (11,3) Distance 4.47213595499958 Setting temper:26.00
Loc : (11,9) Distance 5.656854249492381 Setting temper:26.00
Loc : (11,19) Distance 5.830951894845301 Setting temper:26.00
Loc : (13,1) Distance 7.211102550927978 Setting temper:29.06
Loc : (14,11) Distance 3.0 Setting temper:26.00
Loc : (14,19) Distance 5.0 Setting temper:26.00
Loc : (15,16) Distance 2.23606797749979 Setting temper:26.00
Loc : (16,1) Distance 9.848857801796104 Setting temper:31.79
Loc : (16,3) Distance 9.219544457292887 Setting temper:31.34
Loc : (16,5) Distance 9.0 Setting temper:31.16
Loc : (16,8) Distance 6.324555320336759 Setting temper:27.32
Loc : (17,11) Distance 4.242640687119285 Setting temper:26.00
Loc : (17,14) Distance 3.0 Setting temper:26.00

Loc : (17,16) Distance 3.605551275463989 Setting temper:26.00 Loc : (17,18) Distance 5.0 Setting temper:26.00

Appendix 7： The LSTM net frame we set.(Based on tensorboard)



Appendix 8：Data sets

datanoraw.csv run_.-tag-loss _train.csv run_.-tag-val_loss.csv

You can find them on my GitHub