**Title**

**标题：基于工程化方法的LSTM智能温控系统设计**

**Summary**

我们认为这是一个系统设计问题，从系统设计问题到数学模型，需要抽象。我们选择的抽象方法是基于UML统一建模语言的系统建模方法。对于温度控制系统这样的问题域，我们使用用例图描述用户和温控系统的可能使用场景，然后设计了MVC体系结构框架解决温控系统与用户的交互问题，使用类图描述如何控制温度这个问题域中出现的对象，建立了一个最基本的问题域模型。最终确定要研究的核心问题是在中央处理器上如何综合传感器发来的和用户发来的数据。这个核心问题就是所要研究的数学模型。下面详细阐述从智能温度系统中抽象数学问题的方案。

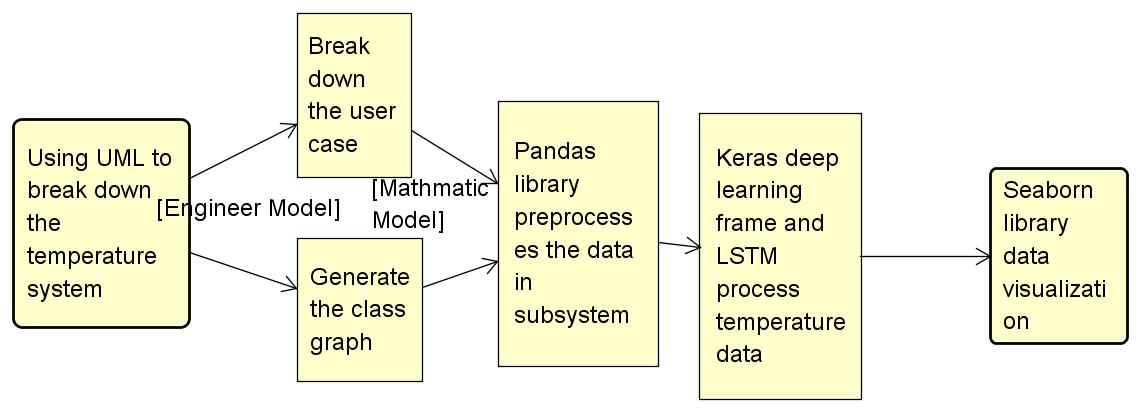


Figure 1工程化建模过程流

1、面向对象分析——用例建模

**用例建模是面向对象分析的第一步。用例包括智能温度系统中的用例以及外部对象，在这个用例图中的外部对象，有用户和中央处理系统两个对象。对象的交互过程是通过用例完成的，用例也就是所谓的场景。用户面对的场景是如何去设定温度程序，换句话说用户去编程控制温度。用户编制的程序应该是能够表征温度的函数，比如说可能是输入温度的函数，然后通过程序，把它翻译成MVC框架中的控制器所能识别的数据流，就可以控制屋内的温度和湿度等信息。对于节省能量、适应用户的喜好这两个用例，在某种程度上存在矛盾，一个系统在运行的过程中必须要先保证用户的满意才能节省能量。用户的喜好判断主要是由系统完成的。有关用户喜好的用例是包含设置温度程序的子用例。用户编程设定的温度场景通过因特网实现，因特网是用户通过自己的计算机，或者是用户的嵌入式终端，如手机设定温度。但是用户编程设定温度非常麻烦，不能总是通过编写程序设计，这样效率极低。因此，我们分析应当设计一个用户可图形编程的界面与用户交互，从而算出温度的变化。这也就是刚才体系结构框架里所述的界面设计。**

**下面将介绍与系统有关的几个用例，这几个用例都能和系统作用感知外部的温度变化，温度变化主要是从传感器中感知，其中有一个扩展用例感知的是湿度的变化，需要湿度传感器。还有一个要感知空气污染水平的传感器。系统除了要从这几个传感器中接收数据，并计算出温度应该怎么变化，还应该能够算出在室内有多少个人，根据人数进一步调整室内温度。例如，如果在一个教室里面，有一百个人，此时空气非常污浊。系统这时要控制空气调节器，开的比较大。但是如果在一个30个人的屋子里，这个空气调节器不用开这么大，因为只有30个人的屋子里面的空气，相对来说比较清新，这也是本系统可以节能的一个原因。**

**综上所述，我们的系统要有一个面向用户的图形界面，能够在界面上进行显式的编程，同时系统也能够根据用户的数据，学习出一个模型，根据用户的人数，根据传感器发出来的各种数据去学习固定的模式。也就是说我们需要据此抽象出数学模型。**

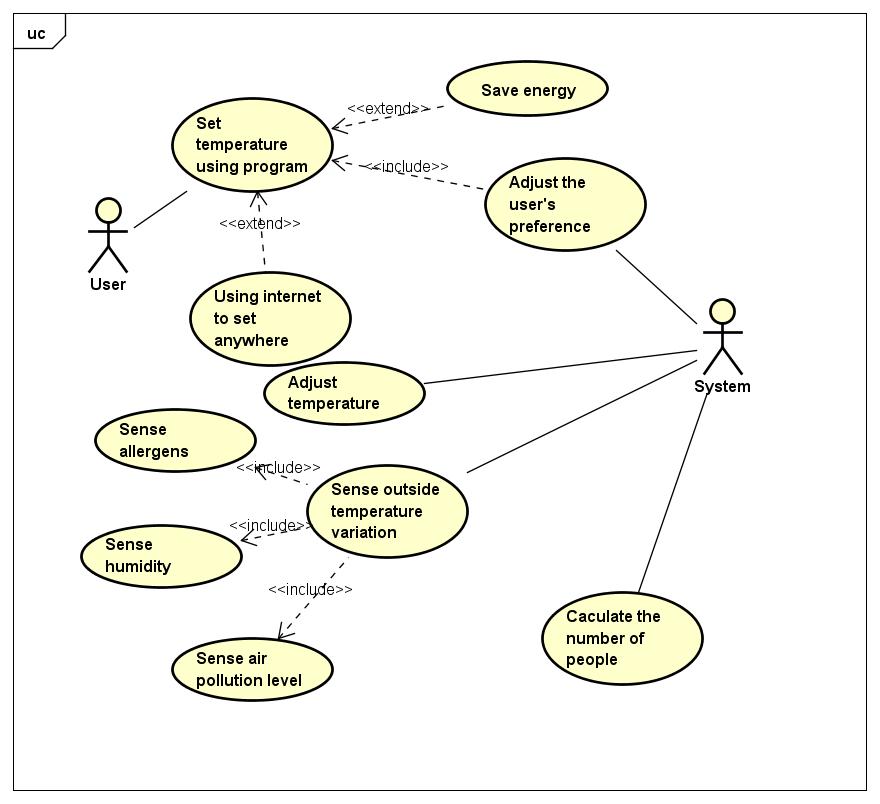


Figure 2温控系统需求分析用例（UML统一建模语言）

2、体系结构框架设计——MVC框架

我们基于MVC设计出一个面向用户和智能温度系统的体系结构框架。MVC框架是一个体系结构框架，由三部分组成：模型对象，视图对象和控制器对象。

其中的控制器对象实例化为温度系统中心处理器。我们选择的是一个核心控制器，它在智能家庭系统中控制所有的任务，并完成核心的计算功能。物联网芯片在智能家居领域中比较广泛，比如树莓派、和英特尔的凌动CPU。他们都可以完成相关的计算工作，并且在神经网络计算棒的支持下奠定了深度学习的神经网络的基础，成为我们下一步选择模型的依据。

视图对象是人机交互的窗口，当用户需要访问智能控制家居系统时，首先需要通过手机应用或者电脑端应用访问系统界面，在下面解释中我们会看到，界面应该具有可编程的特性，以便让用户去图形化编程，控制这个系统的温度。

模型对象具体指LSTM神经网络模型。我们应用Keras深度学习框架实现了这个模型，具体的模型将在后文中讨论，相关的代码以及注释作为附录。

3、温度系统的抽象映射——基于类的建模

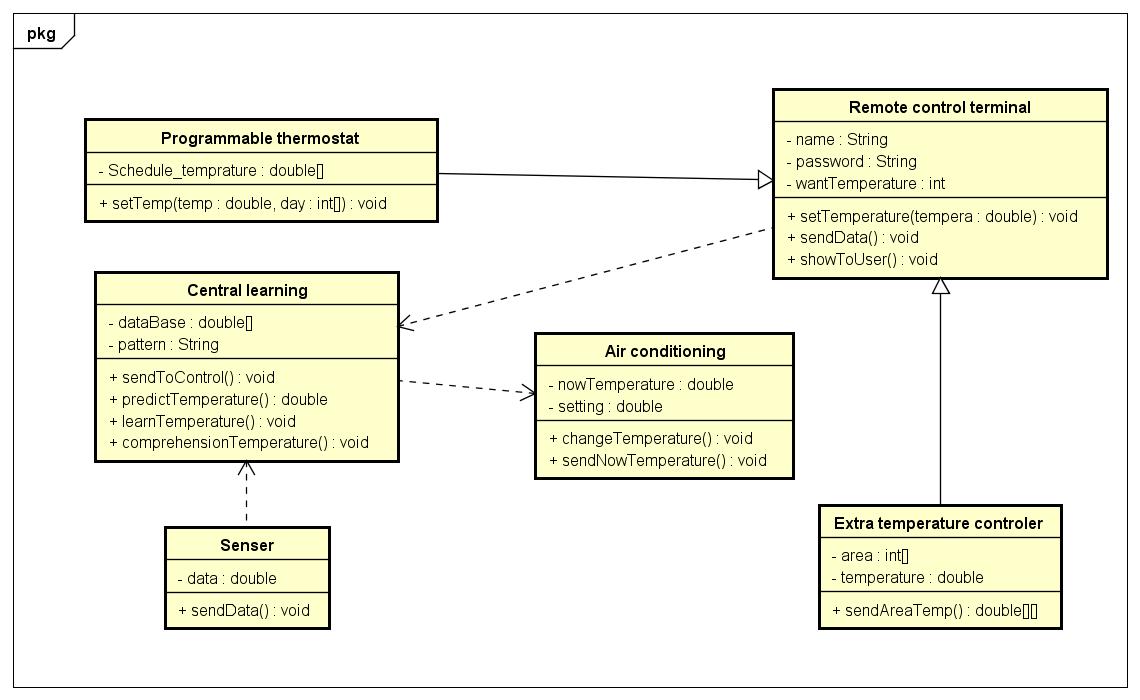
**我们分析出不同的用例，也就是场景。接下来需要根据场景进一步建模，需要进一步需求分析并作系统设计。类图起到了映射的作用，把问题域映射成抽象的对象。绘制出类图，类图中有几个对象呢？程控的温度设定器的实质是上述建立MVC框架中所述的图形温度控制界面。用户还可以在图形界面中输入函数计算，所以也属于程控温度器。这个程控温度器对象中有属性，也有方法。这个方法把这个用户的输入、用户的编程转换成移动数据，发给远程终端。远程终端能够感知到这个消息。远程终端负责接收用户数据，同时负责发送数据。把数据发送给MVC框架中的核心处理器，简而言之是一台服务器。在物联网中设置的服务器不是特别大型的，更多的是嵌入式为主的微服务器。但是考虑到在我们建模的过程中会用到神经网络，在这个系统设计时，处理器必须有USB神经网络加速器接口。因为我们要处理的不是少量数据，而是大量的用户数据。**

Figure 3基于类的建模

**另一方面，我们也要考虑到题目中所给的一些其他限定，只有一个温度调节器和额外温度调节器的情况下，怎么分别对系统产生影响？我们在建模的时候要考虑到，额外的温度控制器对象有继承关系，继承于远程控制终端。在刚才的面向类的建模过程中，我们考虑了在什么样的场景下使用，在这个场景中，在这个问题域中有哪些类？我们需要做些什么？最终把温度系统这样的实际问题归结为数学问题——处理各个传感器和用户发来的数据。在接下来过程中，我们要把这个对工程的建模转化成对数学问题的建模，更详细的讨论细节的数学问题。**

**Contents**

1. **Introduction**
   1. **Background**

In recent years,

* 1. **Problem Statement and Analysis**

In recent years,

1. **Assumption and Symbol Explanation**
   1. **Assumption**

* In recent years,
* In recent years,
  1. **Symbol Explanation**

|  |  |
| --- | --- |
| **Symbol** | **Definition** |
| *A* |  |
| *B* |  |
| *C* |  |
| *D* |  |
| *E* |  |
| *F* |  |
| *G* |  |

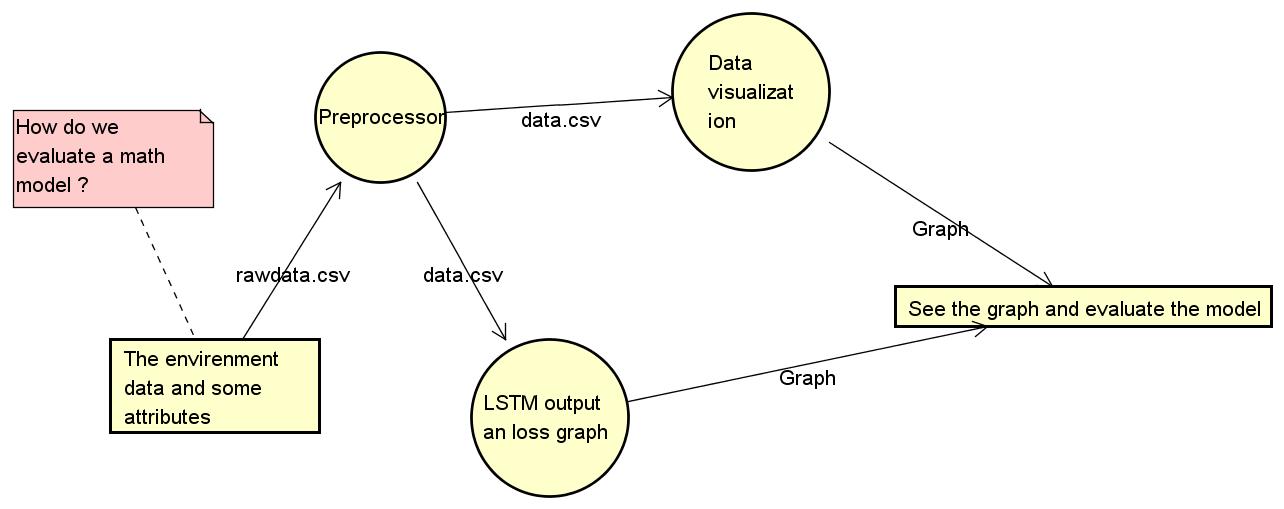
1. **Task 1**
   1. **Data Pre-processing(因为这个题正好用到了数据的预处理，就给它先列出来了)**

数据收集

本文使用深度学习的方法分析数据，因此需要收集较多的数据。收集到的数据有8760条，来自马塞诸塞大学实验室，反映室内室外的环境变化，以及人体的体感温度。数据集共有9个属性，分别为temperature,icon,humidity,apparentTemperature,pressure, windspeed, time, windBearing, dewpoint.通过数据集分析，找出数据间相互作用规律，进而对中心温度控制器建模，最后验证LSTM神经网络模型。

|  |  |
| --- | --- |
| Interpreter of Data field | |
| Temperature | The inside temperature |
| Icon | The weather condition |
| Humidity | The inside humidity |
| apparentTemoerature | Sendible temperature |
| pressure | Stmospheric pressure |
| windspeed | Wind Spees |
| time | Local Time |
| windBearing | Wind direction |
| dewpoint | Dew point temperature (Meteorological academic language) |

数据预处理

下面展现了通过构建数据流图显示需要进行的数据处理。通过对数据可视化图像的分析，找出数据的一般规律，进而利用LSTM神经网络回归数据。数据集存在如数据缺失、存在字符串属性等部分数据不够规范化的问题。

我们使用Python语言的Pandas库和sklearn库进行数据预处理，运行环境是Linux。代码和注释在附录中，分为两个文件解决不同类型的数据正则化问题。

缺失数据的解决方案。缺失数据集中在三列上，缺失量不大，因此我们采用缺失值填充方法而未采用缺失值剔除方法。根据学界有关学者的研究，连续型数据用该属性的平均值去填充，而离散属性使用众数填充数据。鉴于缺失数据全部为连续型的数据，我们通过平均值法填充。

用户数据的填充方案。原始数据集并未提供用户根据具体环境做出的反映，因此我们作为模拟用户尝试填充了小部分调节温度，反映我们自身对于该项环境下的体验。经过80项人工判断填充，我们大致掌握了温度的调节规律。并依规律大规模人工向下填充。经过方案可行性分析，这种人工填充方法模拟了用户的实际行为，能够较好地反映真实系统下用户的有关操作。

非数值属性的编码问题。原始数据中，表现室外天气情况的icon字段出现了{'clear-night', 'wind', 'partly-cloudy-night', 'rain', 'cloudy', 'fog', 'partly-cloudy-day', 'clear-day', 'snow'}这样的字符串。需要将其映射成为整数。考虑到最终用户的输入数据有可能与光线强弱有关，我们按照不同天气情况对应的光线强弱由弱到强依次编码为0-9.为下一步矩阵计算创造数值方面的条件。

时间序列的生成方案。因为数据集提供了时间字段，因此数据预处理时需要根据时间字段排序，以反映时间序列的特征。由于LSTM神经网络是时间序列的网络，我们需要在原始数据的基础上建立滑动窗口。这里我们设置的滑动窗口长度是2，每次窗口滑动一个单位。这里需要特别注意，建立了时间序列后，原始的数据从二维矩阵变换为三维张量。

数据的归一化问题。神经网络的基本原理是把线性不可分的数据映射到高维空间，增强数据的线性可分性。数据归一化可以让数据集中分布在较小的范围内，既可以增强激活函数的非线性映射能力，又可以增强数据集的表现效果。避免数据在经过多层网络时因为计算机浮点运算能力的问题出现梯度消失或者梯度爆炸问题。

数据可视化分析

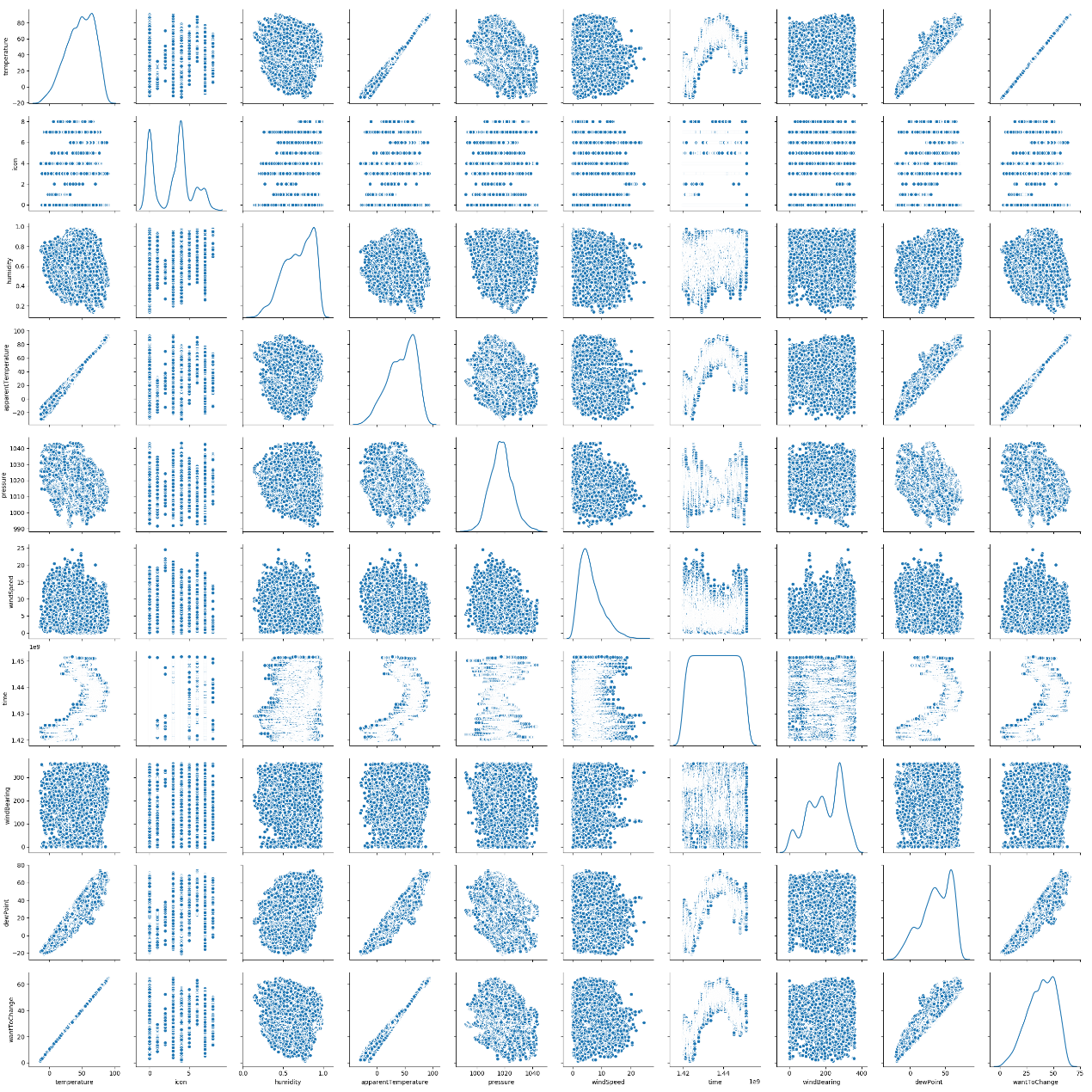
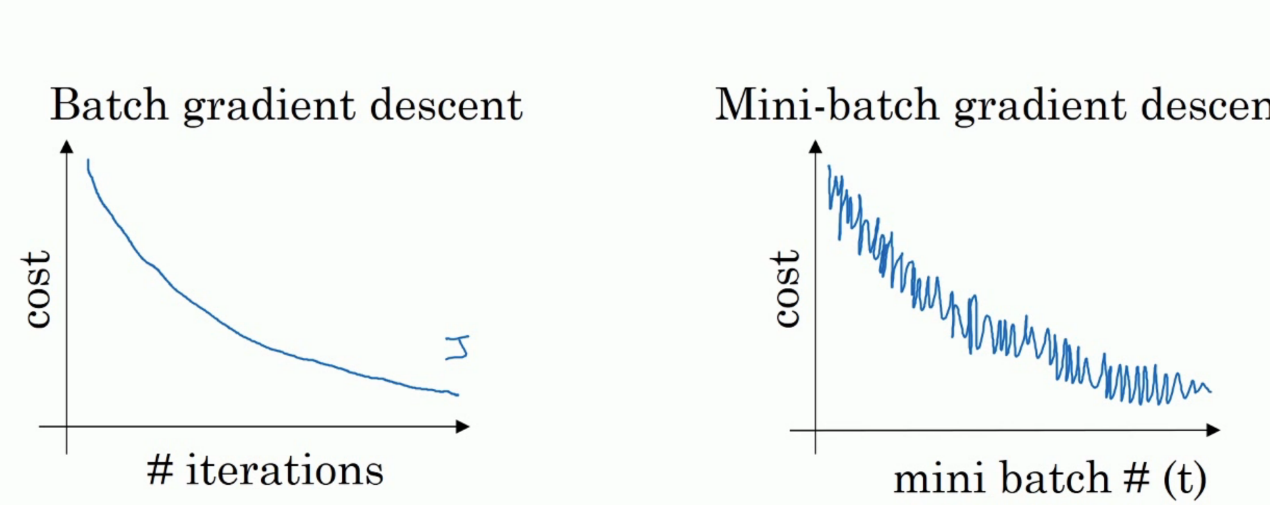


Figure 5 The scatter and distribution Graph

上图展示的是数据集的分布情况，横轴从左到右依次为temperature, icon, humidity, apparentTemperature, pressure, windspeed, time, windBearing, dewpoint, wantToChange.纵轴从上到下与横轴相同。图中对角线上展示的是不同属性的分布情况。可以清晰的发现，数据的分布大多都向正态分布靠近，这也与大数定律的结果相吻合，实验次数越多，分布就向正态分布靠近。

散点图中的time属性具有相当明显的周期性，这说明温度和环境的有关数据和信息都在随着时间的变化呈现周期性的变迁。而温度相关的属性，apparentTemperature, temperature, wantToChange。

LSTM方法建模

数据的正则化方法。为了防止模型过拟合，需要对数据进行正则化处理，这里我们采用数量为20的minibatch方法。如图所示，在不使用minibatch的传统梯度下降算法中，损失函数的下降速率并不规则，这是由于单个梯度下降具有随机性，很难精准的向总体均方误差最小化的方向移动。而minibatch往往具有代表性，能够反映数据总体在误差上的表现。

为了精准比较误差，我们控制LSTM网络返回整个序列而不是一个序列的单个输出。

模型选择方面，全连接型神经网络和卷积神经网络不适合。卷积神经网络一般是用在图像的处理里面，解决高维数据压缩问题，但是这个数据集维度仅为9个属性。全连接神经网络无法很好的处理时间序列问题。朴素贝叶斯方法对于样本数量较少的情况下拟合效果好，但是缺乏时间序列的判断性。而RNN循环神经网络，能够表现时间序列的变化，但是因为缺乏控制流。因此不能较好地对长时数据建模。

下面是使用LSTM的网络结构与参数：

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=========================================

lstm\_1 (LSTM) (None, 2, 4) 208

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 2, 1) 5

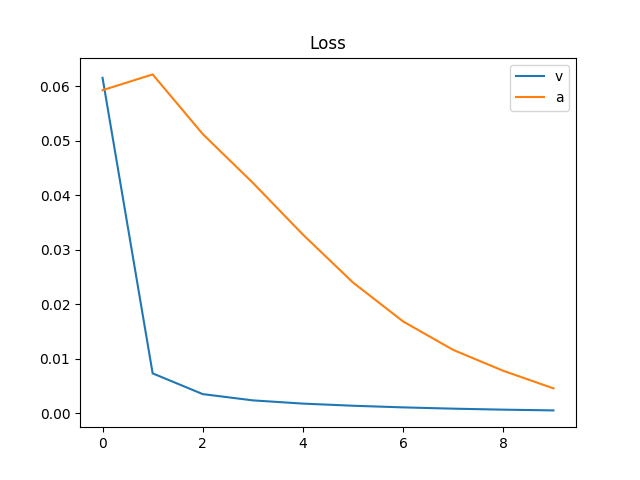
=========================================

Total params: 213

Trainable params: 213

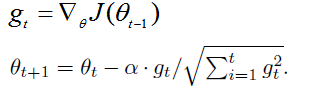
Non-trainable params: 0

LSTM网络中，我们使用sigmoid函数作为遗忘门的控制函数，因为sigmoid函数的值域是在0到1之间，因此能够把数据，而真正的数据使用tanh函数激活。接下来我将阐述我们数据预处理以及利用专业模型进行建模的细节和算法的细节。



图中所示为模型拟合过程中的损失下降曲线，可以看出，使用minibatch方法，在迭代少于10次时下降效果较好，v-validation set，a-train set。

因为我们现在并不知道用户是多少，接下来我们去用自适应方法优化这个模型，使用传统的均方误差，作为损失函数。使用的是Adagrad优化器。



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

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  raw\_data=pd.read\_csv("datar.csv")  temp=set(raw\_data['icon'])  print(temp)  size\_mapping={label:index for index, label in enumerate(temp)}  raw\_data['icon']=raw\_data['icon'].map(size\_mapping)  sigma=0.2  mu=0  wanttochange=0.607\*raw\_data['temperature'].values+10.092+np.random.normal(mu,sigma,raw\_data.shape[0])  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())  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)

|  |
| --- |
| from matplotlib import pyplot as plt  from pandas import read\_csv  import math  from keras.models import Sequential  from keras.layers import Dense  from keras.layers import LSTM  from sklearn.preprocessing import MinMaxScaler  import numpy as np  look\_back=2  epochs=10  batch\_size=20  scaler\_x = MinMaxScaler()  scaler\_y=MinMaxScaler()  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  data\_set = read\_csv('datanoraw.csv', header=0, index\_col=0)  data\_set=data\_set.values.astype('float32')  data=data\_set  scaler=MinMaxScaler()  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)  #X\_train = np.reshape(X\_train, (X\_train.shape[0], look\_back, X\_train.shape[1]))  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')  history=model.fit(X\_train, y\_train, epochs=epochs, batch\_size=batch\_size, verbose=2,validation\_split=0.33)  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') |

**Data Collection**

In recent years,

* + 1. **Data Filling**

In recent years,

* 1. **Model Name**

**这里写模型的介绍，作用，具体的方程，图像等。。**

1. **Task 2**

**这里写模型的介绍，作用，具体的方程，图像等。。**

1. **Task 3**

**这里写模型的介绍，作用，具体的方程，图像等。。**

1. **Strengths and Weaknesses**
   1. **Strengths**

* In recent years,
* In recent years,
* In recent years,
  1. **Weaknesses**
* In recent years,
* In recent years,
* In recent years,

**References**

[1] COMAP. (2017). ICM Problems.zip. http://www.comap.com/undergraduate/contests/mcm/contests/2017/problems/2017\_ MCM-ICM\_Problems.zip

[2] EPA, “Smart Growth: A Guide to Developing and Implementing Greenhouse Gas Reductions Programs.” 2011. http://www.sustainablecitiesinstitute.org/Documents/SCI/Report\_Guide/Guide\_EPA\_SmartGrowth GHGReduction\_2011.pdf