View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by UT Digital Repository

Copyright

by

Ahsen Dinc

2018

The Thesis Committee for Ahsen Dinc
Certifies that this is the approved version of the following Thesis:

Adaptive Learning Approaches for Smart Home Environments with a Simulator Implementation

APPROVED BY
SUPERVISING COMMITTEE:

Christine Julien, Supervisor

Edison Thomaz

Adaptive Learning Approaches for Smart Home Environments with a Simulator Implementation

by

Ahsen Dinc

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

May 2018

Dedicated to my loving family.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Christine Julien for providing me a great research opportunity, for her critical reviews and constructive feedbacks, and also for her support and patience in drafting this thesis. I would also like to thank Dr. Edison Thomaz for his patience and time to review my thesis.

I would like to thank my precious parents, Serpil Dinc and Muammer Dinc for their continuous support, love and trust on me throughout my life. I couldn't have done any of my accomplishments without feeling them always by my side. I would like to also thank my sisters Asli Dinc and Puren Dinc for their love and support.

Last but not least, I would like to thank Kaan Ege Ozgun for always being there for me and supporting me throughout this journey.

Abstract

Adaptive Learning Approaches for Smart Home Environments with a Simulator Implementation

Ahsen Dinc, M.S.E

The University of Texas at Austin, 2018

Supervisor: Christine Julien

Smart home solutions are utilized in a different way by each user according to the user's unique needs and preferences over his unique home setting. User – device interactions themselves carry some implicit characteristics of the context the smart devices are used. In order to better exploit Internet of Things technologies in smart home environments, the user's interaction history can be leveraged to generate knowledge of his behavior habits. With a purpose of achieving personalized decision making of smart lighting environments, this thesis presents three learning model approaches, which are based solely on the individual's interaction habits with the devices, adaptive to changes in inhabitant's device usage behavior, and do not require preset data for initialization. Individual interactions with smart light devices are contextualized based on the timestamp of the interaction, and ambient light intensity reading of the room at that instant. Moreover, the thesis introduces a Java simulator, which interactively demonstrates the behavior of a personalized smart home setting with the integrated learning models with a feedback mechanism. The simulator leads up to a way of

evaluating adaptive learning models in smart home environments, reducing the immediate need of testing their behavior in a real life smart home, which is both hard and costly to construct and maintain. The learning approaches, namely K-Nearest Neighbor and Softmax Regression with batch learning and online learning algorithms, are evaluated with different datasets representing different scenarios, and the results show that all three methods are able to perfectly capture the usage pattern when the interactions with distinct “things”, smart light devices, are separable in terms of their corresponding context. For more complex datasets, which have overlaps between the usage context of distinct devices and big changes in user behavior over time, the online learning algorithm needs more data in order to catch the performance of KNN and Softmax Regression with batch learning algorithms.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

As devices are increasingly connected, it has become inevitable to hear the concept, The Internet of Things. The conversation on The Internet of Things has been growing for decades, coming from both academia and industry. What exactly is this concept of The Internet of Things and what is its impact on our lives? The Internet of Things is considered as the Future Internet, where all devices and people are connected anywhere, any time. It entails the connectivity of devices such as home appliances, vehicles, mobile phones, physical devices and other items with sensors, actuators, electronics and software, and the ability to exchange data. [1] Here, the term "thing" can stand for a real or physical device as well as a digital or virtual entity, that exists and can be identified by identification numbers, names or location addresses through a network or a path.

As a result of the rise of distributed systems on the cloud, physical devices in the environment, and even human beings, Internet of Things technologies have already become an indispensable part of our lives in a very smooth way. Almost everybody carries a smart phone, which is equipped by many sensors and different network capabilities. IoT technologies are entering into our daily lives at home as well; smart homes are introduced with more and more sensors and actuators, forming new opportunities for smart home automation. Analyzing the data captured from these smart

devices and providing intelligent services to the household based on this data analysis has become a popular research area.

1.1 Problem Statement

As smart homes become equipped with more and more devices, these advancements in smart environments bring complications to users in the form of device interactions. For example, in today's smart lighting systems, there can be many controllable lights in a house and the user may have a hard time navigating a technology that is not familiar, simply for the purpose of turning on the desired light. While the user just wants to turn on a light after entering a room, there is not so straightforward interaction flow in the digitally enabled environment. First, an app must connect to a "bridge" that gives a list of available lights, then the user must select a light by name or identifier specific to the bridge, and manually reselect a light if he chooses incorrectly. However, in order to increase the adoption of smart environments by the average user, the digitally enabled world should instead ease the user's life with more intuitive interactions by implicitly understanding and responding to a user's unique habits and preferences.

1.2 Purpose and Contribution

Every user has his or her own way of using each appliance at home, as every family has a different life style and different needs. As user's interact with individual

devices, the device can create a trace of the user's unique behavior; extracting the user's interaction habits towards the available devices can influence decisions in human-centered smart environments. *The purpose of this work is to improve the effective use of Internet of Things technologies by leveraging user interactions with smart devices, ultimately providing the user personalized smart spaces that are adaptive to the individual's unique behavior habits and changes*.

As the user interacts with devices in the environment, these interactions create non-negligible amount of data about individual usage of the devices. In order to better exploit Internet of Things technologies in smart home environments, this interaction history can be leveraged to generate knowledge of behavior habits of the individuals. To achieve personalized decision making of smart lighting environments, this thesis presents three learning model approaches, which are based solely on the individual's interaction habits with the devices and adaptive to changes in inhabitant's device usage behavior. These approaches do not require preset data for initialization. These three learning methods, K-Nearest Neighbor, Softmax Regression with online learning, and Softmax Regression with batch learning, capture the unique pattern of device interactions of the individual and contribute to decision making of the system, choosing the right light under a specific context for a specific user. The feature extraction of the models is based on date and time of the attempted turn on action of the user over smart light devices, accompanied by the ambient light intensity sensor reading of the room that the user is in. Moreover, an interactive Java simulator with a Machine Learning stack is implemented,

which demonstrates user interactions in a personalized home setting with the proposed learning models embedded. Through navigating an actor and interacting with the available devices in the virtual home, the approach builds an interaction history for the user in the virtual environment and evaluates the integrated learning models; the simulator thus serves as a proxy for the immediate need of testing the approaches in real life smart homes, which is both costly and hard to construct and maintain.

Chapter 2: Related Work

This section is dedicated to the literature review related to this research.

2.1. SMART HOME DATASETS

This part of the thesis reviews some of the previous work on smart home systems that present popular open source smart home datasets. This review of available smart home datasets is particularly important because user - device interaction datasets are needed to evaluate the performance of three learning methods to be proposed in this thesis, and therefore, I am in search of available real smart home datasets capturing user's interactions with smart environments.

2.1.1 CASAS: A Smart Home in a Box

CASAS [2] specifies the principal components of smart home system such as sensors and actuators. It includes different types of sensors so that activities of daily living (ADL) can be recognized using the data from these sensors embedded in the environment. Apart from construction of smart home systems, CASAS [2] also tries to solve activity recognition problem to predict daily activities of the user. For activity recognition, CASAS [2] classifies the most recent sensor events using Support Vector Machines, Hidden Markov Models, and Conditional Random Fields. The time of the event is also used as a feature in the classification problem; time is discretized by

splitting the hours of a day into four equal time intervals. Another learning problem CASAS [2] addresses activity discovery from unlabeled sensor data, and the corresponding solution is an unsupervised algorithm that has been applied to extract information about activity patterns. This algorithm finds sequence patterns in the dataset. Activity discovery algorithms are applied onto this reduced dataset, and activities (such as cooking, eating, sleeping, etc.) are then identified from the dataset. Activity discovery algorithms help detect activities in which the users spend time on secondary activities other than their main activity. By detecting these deviations, activity discovery prevents some instances being labeled as unidentified, hence increasing the accuracy of activity recognition.

Overall, CASAS [2] focuses on the construction of real life smart home systems and recognition of the user's activities of daily living based on the collected sensor data. The datasets generated through CASAS [2] are very popular among smart home research groups. However, the collected data is specifically useful for ADL events; our work requires datasets that contain time-stamped user interactions with lighting devices accompanied with ambient information like light intensity sensor readings so CASAS [2] datasets were not suitable to be used for the evaluation of my interaction based learning models. However, one approach in CASAS [2] that inspires my work is the discretization of the hours of a day to use in feature extraction. In CASAS [2], hours of the days are divided into four intervals while I use ten intervals to achieve better resolution on the time feature compared to that of CASAS [2].

2.1.2 ARAS Human Activity Datasets in Multiple Homes with Multiple Residents

ARAS [3] provides a real world human activity dataset, which is collected from two houses over one month. The collected dataset is more realistic compared to traditional datasets since it contains activities of humans in real life instead of activities that have been simulated with laboratory experiments. The dataset covers a large number of activities and is open for public use. The data are acquired from 20 different sensors which communicate via the ZigBee wireless protocol. The sensors include force sensors, photo cells, contact sensors, proximity sensors, infrared receivers, and temperature sensors. The actions detected include sleeping, sitting, cooking, watching TV, opening and closing doors. Using this dataset, probabilistic machine learning models are employed for activity modeling. Since activity data is sequential, Hidden Markov Models have been applied, and average accuracy of 61.5% and 76.2% for each house. However, as in CASAS [2], the generated dataset also focuses on the activities of daily life, instead of user-device interactions for smart lighting systems.

2.2 SMART HOME SIMULATORS

This part of the related work section goes over some of the existing smart home simulation tools for synthetic data generation of activities in smart homes.

2.2.1 Simulation of a Smart Home Environment

Ariani et al. [4] propose a smart home simulation tool that captures inhabitant interactions leveraging ambient sensors. In the tool, researchers interact with a map editor to design a smart home floor plan, with the ability to draw shapes on a 2D canvas. After designing the floor plan, the virtual home can be equipped with the ambient sensors according to researchers' choices. Binary motion detectors and binary pressure sensors can be simulated, leading to simulated inhabitant movements. However, the simulated activities of the residents are limited by the movements, and there is no data generation capability of user interactions with smart device.

2.2.2 SIMACT: A 3D Open Source Smart Home Simulator for Activity Recognition

SIMACT [5] presents a 3D smart home simulator which is implemented Java with the purpose of helping researchers in the activity recognition field by providing a platform to perform experiments related to detecting activities of daily life. SIMACT [5] uses a model-based approach for simulations, where pre-defined models are used to generate synthetic data. SIMACT [5] includes pre-recorded scenarios captured from clinical experiments, and using these scenarios, offers its users the ability to generate datasets for the recognition of activities daily life. However, this simulator renders only contact sensors, it detects if an object is touched and stores this information. There is no available way of generating interaction between users and smart devices.

Chapter 3: Approach

Through the advancement in mobile devices and sensor and control systems, the way we live is being changed. Home automation is a really hot topic. It is now possible to manage home appliances using a smart phone, assuming the appliances are able to be discovered and exchange data. There is much work in both academia and industry on remote control systems for smart homes, where the user chooses the correct device to do a certain job. [6, 7, 8] One of the industry solutions in smart home market is Philips hue, which focuses on wireless lighting systems and offers its users full control over their personal lighting system via a smart phone or a voice control device. [9] Another product in the market is the Wink hub, which introduces a way to wirelessly connect smart devices from multiple manufacturers, including smart light bulbs, and provides the user remote control over these devices via a smartphone. [10] However, in all these solutions the user must navigate a list of lights connected to the central hub, and choose the light that he or she wants to turn on among the many devices in the list without any proactive decision making from smart device solution sides.

Figure 3.1: Smart House Control Illustration

Smart home solutions are utilized in a different way by each user according to the user's unique needs and preferences over his unique home setting. User – device interactions themselves carry some implicit characteristics of the context the smart device is used. Leveraging the user's device interaction history, personal device usage characteristics can be unveiled paving the way for transitioning into more proactive smart environments. This research focuses on capturing the pattern of the specific user's interactions with a set of "things", leading to predict the behavior of the user without the user specifying it and to make a decision accordingly. Main approach for this research is to come up with an adaptive self learning and prediction model based on the user's past interaction history on specific types of remotely controllable appliances in the environment such as smart light devices exploiting straightforward Machine Learning

algorithms while considering physical scope of the environment. It is important for the model to be personalized per user since every user has his own way of using devices as well as a different device setup in different rooms at home. Also, the user's needs may change over time and this change should be reflected in the model. For this purpose, a feedback mechanism is integrated in the proposed solution in order to make the model adaptive to the user's habits.

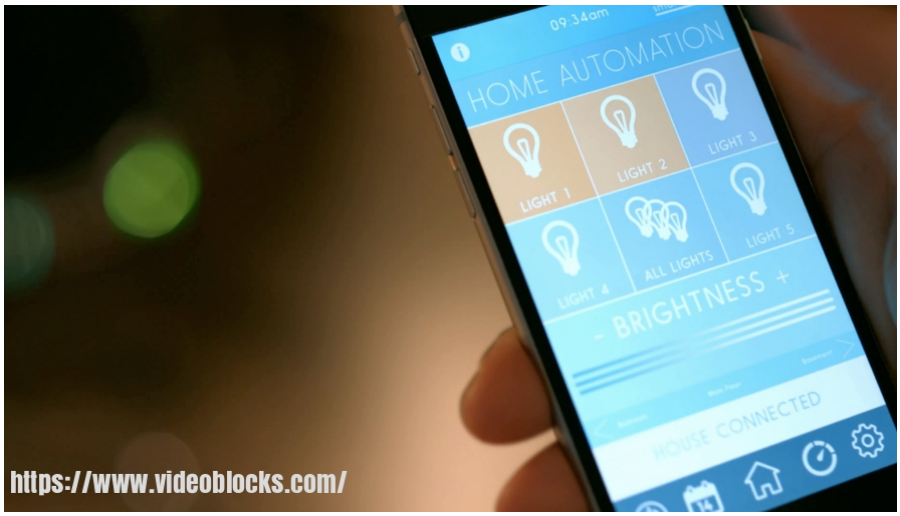

Figure 3.2: Remote Home Control via Smart Phone

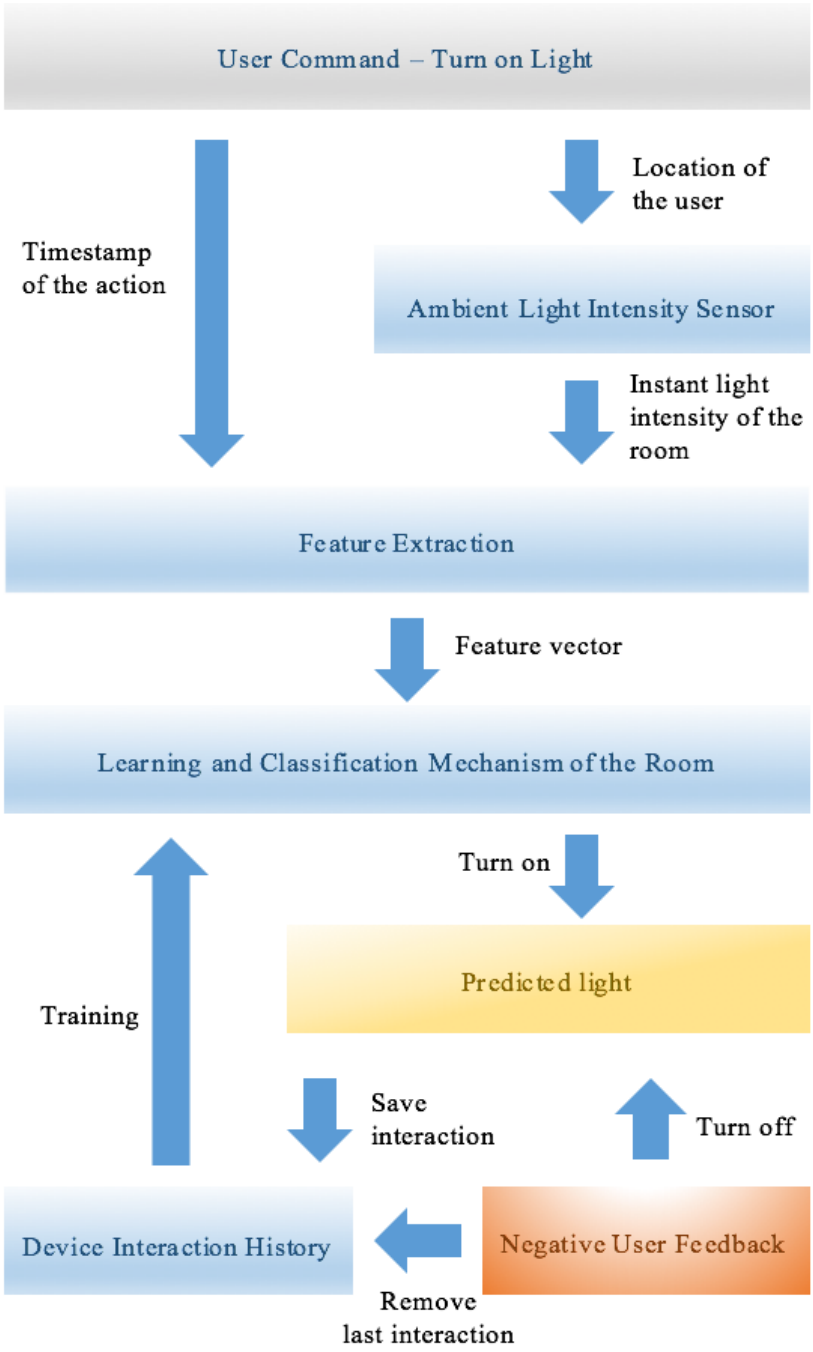

Figure 3.3: System Architecture Diagram

A key point to highlight is that the model I developed does not require any prior pre-tuned data; instead it is trained solely based on the individual user's interactions with the smart devices. Starting from day one, the interactions of the user with the physical device are captured; as new interaction data arrives, the model is updated accordingly so that any change in the user behavior is reflected in the model, leading to better informed decision making. The learning and decision making mechanism proposed in the work is implemented in a simulator for light control in smart home environments. The simulator demonstrates a smart home with different lights at different locations and provides a real-time sense of effective personalized smart light control that leverages the learning and decision making design in this work, which will be introduced in the next section.

3.1 Design of The Proposed Learning and Decision Making Model

Leveraging individuals' behavior routines in the context of smart devices can lead to better decision making in human-centered environments. This work is focused on adapting the smart spaces according to the interactions of distinct users with devices in such a way that the personalization is done seamlessly and open to feedback from the user in case of any behavior changes. The implementation of the model is done for smart lighting systems.

https://www.marsdd.com

Figure 3.4: Smart Lighting Illustration at Home

At this point, it would be good to give an example scenario in order to make more clarifications on the reasoning. Bob leaves home around 8 am to go to work and comes back home around 6 pm. Let's have a look at his schedule for a day.

- Bob wakes up around 7 am.
- He enters the bathroom to take a shower and turns on the light A in the bathroom.
- Before leaving home in the morning, he enters the kitchen and turns on the light B around 7.30 am, in order to prepare breakfast. Then, he leaves the apartment to go to work.
- He comes back home around 6 pm.
- He enters the bedroom to change his clothes and turns on light C in the bedroom.

- He goes to the kitchen around 7 pm to prepare dinner and turns on light A. He realizes the light level is not enough because the sun is already set, so he also turns on light D.
- He enters to the living room around 7.30 pm to watch TV and turns on light E.
- He enters to the bedroom to sleep around 11 pm, turning on light F, which is a small bedside lamp.

All these interactions of Bob with the light devices at home are valuable in the sense that they can be used to determine his needs and preferences in different situations. Our system automatically determines the "best" light to illuminate based on the user's current environment and the collected past interactions of the user in that environment. Each room is evaluated individually since different rooms often entail different lighting needs, and have different light positioning and specifications.

My work characterizes interactions with light devices in a smart human-centered space based on four distinct features: the location of the device, calendar date of the interaction, time of day of the interaction, and the ambient context of the interaction (i.e., in this work, a reading from a light intensity sensor).

3.1.1 Physical scope and location of the device

Things, both sensors and actuators, have physical scopes in such a way that a sensor has an area about which it can collect information while an actuator has an area that is able to have an impact on. It is essential to embrace this physical scope factor in

order to make personalized use of an Internet of Things infrastructure. Physical limitations like walls should be taken into account in order to come up with a more realistic model. For this reason, this work with lighting devices leverages the concept of a room, where each room has its own model, and this model trained individually based on the data for the devices encapsulated by the room.

The location of the user is also important to determine the room in which the particular interaction is happening. Every light device belongs to a room, and knowledge of which room the actor is in at the time of the interaction helps eliminate the lights that are not in the relevant physical scope. There is ongoing research on indoor localization technologies [11, 12] that can be leveraged for the real life implementation of this work.

3.1.2 Calendar date of the interaction with device

Every place has its own characteristics for weather and seasons. Duration and strength of daylight at a place changes throughout the year as a result of the axial tilt of the Earth. Seasons are formed and concepts like longest, shortest and equinox days of the year emerge, e.g. for the northern hemisphere, the 21st of June is the longest day of the year, while the 21st of December is the shortest day; in the southern hemisphere it is vice versa. The time for sunrise and sunset changes throughout the year. These differences in daylight should be reflected in the model since the user's behavior may behavior change across different seasons.

Also, the user's life style may change season to season. For example, for the scenario with Bob, Bob may work longer times during the summer so his arrival to home may be later than that of the winter.

For these factors, the calendar date has an effect on the learning model; my approach reflects this by dividing the year into four different categories representing four seasons and then associating the interaction date with one of the four seasons.

3.1.3 Time of the interaction with device

A major indicator of the regularity of interactions is the time of day. As in the example of Bob, there are some times the user is more likely to interact with a device in the room rather than another one. Knowledge of the time of day when the interaction occurs helps build a model based on the user's daily routine. For this reason, my learning model represents the 24 hours of the day in ten categories. Each category represents different time slots, where the current interaction falls into only one slot. The categories are sized differently considering the sleep time of the people and the daylight strength during the day. The time dependence of the model will be explained in detail in the implementation section of the paper.

3.1.4 Complementary sensor reading for the specific device type

While the time and date dependence have a strong foundation in the model, the weather may cause a user's interactions to deviate from the seasonal expectations during

the day. These changes may be manifest in their effect on the light intensity of the room and result in different needs for the user. Having the knowledge on the instantaneous light intensity reading of the room contributes a model that is more robust to abnormal fluctuations in the routine.

Moreover, the light intensity in the room directly affects the user's choice on the number of lights. In a scenario where a single light is not sufficient to fulfill the user's desire, the light intensity can be helpful in that, when the user turns on a light in the room, it is expected that the light intensity in the room will increase, resulting in a change in the light intensity sensor reading. When the user needs an additional light to be turned on and commands our system to turn on the light again, this change in the sensor reading eliminates the effect of overlapping date and time intervals of different lights, and leads to better decision making. Therefore, our system is able not only to capture the right device to turn on from the user's perspective but also to preserve the preferences over the sequence of the actions, turning on multiple lights.

Chapter 4: Implementation

This chapter is dedicated to the implementation part of the work and divided into four parts: data generation, data pre-processing and feature extraction, machine learning algorithms, and simulator implementation.

4.1 Synthetic Data Generation

It is not a surprise that development of smart home designs is a popular research domain considering that the home is an essential part of human beings. Data is the key requirement for the advancement of the research in learning for not only smart home but for all Internet of Things systems. Representative datasets are inseparable part from this vision of smart home environments, and they play a vital role in training, validating, and evaluating emerging machine learning models for smart homes. As in this work, both the training and validation of the machine techniques that capture the behavior of the user and make predictions require datasets of relevant smart home scenarios. However, there is a lack of real smart home datasets, due to the excessive cost of construction of real smart homes. Privacy and sensitivity of the data is another limiting factor against the open source real time smart home datasets. Yet, there are some open source datasets available, which include many sensor readings from within the house [2, 3], but these datasets focus on the activities of daily life instead of time-stamped interactions between

users and devices. As discussed previously, in order to enable personalized smart home interactions, our system is based on the user's device interaction history, consisting of the location of smart devices that the user interacts with, as well as the date, time and the instantaneous light intensity sensor reading for each interaction. Due to the lack of suitable real smart home interaction data, we generate synthetic data as a substitute for real data, representing different scenarios. This generated data is used to validate the learning models, which will be explained in the upcoming sections.

4.2 DATA PRE-PROCESSING AND FEATURE EXTRACTION

There are three hundred and sixty-five days in a year, and representing every single day as an independent feature would increase the dimensionality of the feature vector significantly. Further, it may actually be detrimental to the model since different days may exhibit similar behavior patterns. Therefore, our approach leverages a higher level discrete quantization that divides into categories in such a way that each date is represented by its corresponding season like Winter, Summer, Spring, and Fall. A one hot encoded vector with size four is formed based on these four seasons, where the only true bit is at the index of the corresponding season.

We use a similar approach for the time vector in order to obtain a reasonably-sized feature vector that sufficiently expresses the data. According to an average user action schedule at home, the time intervals are formed also considering the possibility of similar environmental situations such as light intensity. For example, every human being

goes to sleep so the most likely times for the user to be asleep are grouped together and represented in the same chunk. Again, a one hot encoded vector with size ten is formed based on the ten different time slots; only one bit among the ten bits is true at the position where the corresponding time slot falls in.

The last characteristic exploited is the instantaneous light intensity sensor reading of the room. The light intensity in the room directly affects the user's desires relative to turning on smart lights. In a scenario where a single light is not sufficient to fulfill the user's desire, the light intensity can be helpful in such a way that when the user turns on a light in the room, it is expected the light intensity in the room to increase, resulting in a change in the light intensity sensor reading. When the user needs an additional light to be turned on and commands our system to turn on the light again, this change in the sensor reading eliminates the effect of overlapping date and time, and leads to a better decision making. Therefore, our system is able not only to capture the right device to turn on from the user's perspective but also to preserve the preferences over the sequence of the actions, turning on multiple lights.

4.3 MACHINE LEARNING ALGORITHMS

The main purpose of this work is to come up with a personalized smart environment implementation that captures the user behavior patterns for turning on lights in a smart home. Starting without any prior data, our system incrementally trains itself based on the user's interaction history and feedbacks. We implement the K-Nearest

Neighbor algorithm and the Softmax Regression algorithm with both batch learning and online learning. We implement the algorithms in Java without using any external libraries for future compatibility purposes with Android phone applications.

4.3.1 K-Nearest Neighbor Algorithm

The K-Nearest Neighbor algorithm (KNN) is a non-parametric supervised algorithm, which relies on the geometric distance among the data points. Here, k represents the number of neighbors nearest to the point of interest. The classification is based on the majority vote over the labels of the neighboring data points. A test point is assigned a label based on the most common label of the closest k neighbors. KNN algorithms can use different distance metrics to determine the neighbors nearest to the point to be classified; we use the Euclidean distance.

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Figure 4.1: Euclidean Distance Formula

The training dataset consists of vectors in the multidimensional feature space and their assigned labels. In the classification stage, for a test data point, the distance between this data point and each training data sample is calculated. The closest k data samples,

where k is a hyper parameter and chosen heuristically, are found. For the list of neighbors, the majority vote algorithm is leveraged over their labels in order to make the final classification.

KNN does not require any parameter and explicit training phase. This nature of KNN makes it suitable to be analyzed for this research since zero initialization is a requirement for the system.

4.3.2 Softmax Regression

Softmax Regression, also called Multinomial Logistic Regression, is a supervised algorithm for multi-class classification through a generalization of the Logistic Regression algorithm. We first explain the Logistic Regression algorithm, which handles only the binary classification problems.

Logistic Regression focuses on the binary classification problem, in which the label takes only binary values. It uses a logistic (sigmoid) function to predict the probability that the data sample belongs to the class labeled as 1. The logistic sigmoid function can be found in the equation below:

$$P(y = 1|x) = h_\theta(x) = \frac{1}{1 + e^{-(\theta^T x + b)}}$$

Figure 4.2: Sigmoid Function

Logistic Regression is a generalized linear model. Predictors of logistic regression can be written as a function of a parameter $\theta^T x$, which is a linear function of x. In this notation, θ refers to the weight vector and b refers to the bias. In the upcoming equations, will will use a concatenated notation of weights. In this notation, the weight vector is extended by the bias term and the data vector is concatenated with 1 so that the overall inner product of these vectors will produce $\theta^T x + b$.

Logistic Regression tries to maximize the log likelihood of the training examples, whose probabilities are modeled by the sigmoid function given in the formula above. The probability that the data sample belongs to class 1 is represented as $h_\theta(x)$; the probability that it belongs to class 0 is $1 - h_\theta(x)$.

$$
\begin{aligned}
L(\theta) &= P(y|x;\theta) \\
&= \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta) \\
&= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y(i)} (1 - h_\theta(x^i))^{1-y^{(i)}}
\end{aligned}
$$

Figure 4.3: Linear Loss Function

Since the logarithm is a monotonic function and taking the gradient of the loss function in the equation above is a hard problem, we exploit the logarithmic loss function in the implementation.

$$
\begin{aligned}
l(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))
\end{aligned}
$$

Figure 4.4: Log loss Function

Parameters of the classifier are updated using gradient-ascent optimization over the direction of gradient of log likelihood of the training samples.

Unlike logistic regression, in Softmax Regression, the class probability of each data sample is modeled as a softmax function, which can be considered a generalized version of a sigmoid function onto multiple classes. The formula for the softmax function can be seen in Equation 4. Softmax ensures that the probability distribution is formed over the output of all the classes. The Softmax Regression algorithm uses a cross entropy loss function, which can be seen as a generalized version of the log loss function of the logistic regression. [13]

$$
P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T} \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T} \mathbf{w}_k}}
$$

Figure 4.5: Softmax Function

$$
\begin{aligned}
J(\theta) &= -\sum_{i=1}^{m}\sum_{k=1}^{K} \mathbb{1}\{y^{(i)} = k\} \log \frac{\exp(\theta^{(k)T} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)T} x^{(i)})} \\
&= -\sum_{i=1}^{m} (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) + y^{(i)} \log h_\theta(x^{(i)}) \\
&= -\sum_{i=1}^{m}\sum_{k=0}^{1} \mathbb{1}\{y^{(i)} = k\} \log P(y^{(i)} = k | x^{(i)}; \theta)
\end{aligned}
$$

Figure 4.6: Loss Function of Softmax Regression

Basically, softmax regression tries to minimize the loss function given above. However, this function does not have a closed form solution. Therefore, we use Gradient Descent optimization to converge to the global minimum. To do that, we need to calculate the gradient of the loss function with respect to the weight vectors and bias terms.

$$
\nabla_{\theta^{(k)}} J(\theta) = -\sum_{i=1}^{m} x^{(i)} (\mathbb{1}\{y^{(i)} = k\} - P(y^{(i)} = k | x^{(i)}; \theta))
$$

Figure 4.7: Gradient of the Loss Function for Softmax Regression

Batch learning: In batch learning, gradients are calculated using all of the data points, with Batch Gradient Descent optimization. Since all available data is taken into account, gradient updates are slower than other types of learning methods such as online learning and mini-batch learning.

Online learning: In online learning, each training sample can be considered as a batch by themselves. We can resemble this as taking a batch size of 1 in such a way that as a new data point comes (e.g., as a result of the user's turning on a light), parameters of weight vectors and bias terms are updated based on the incoming data point. Stochastic Gradient Descent optimization is used in the online learning; gradients are calculated based on only one sample, and as a result they will be noisy compared to batch learning.

We use a learning rate (step size) of 0.01 for both Stochastic Gradient Descent and Batch Gradient Descent optimization. In order to avoid over-fitting, we apply L2 regularization onto weight vectors, where the regularization rate is 0.001. Weights are initialized with a Gaussian random distribution with zero mean and 0.01 standard deviation. The bias terms are initialized with zeroes.

4.4 SIMULATOR IMPLEMENTATION

This work also implements a Java Simulator to demonstrate the system working in a typical home setting. Within the simulator an actor, or in real life the actor's device (e.g., smartphone), interacts with an environment containing *things*, such as smart light switches. As the actor discovers and interacts with these things, the simulated smart home environment learns how to behave according to the instantaneous context of the actor. The simulator exploits our KNN and Softmax Regression algorithms to learn the user preferences in a particular context and adapt the environment accordingly. For both KNN and Softmax Regression, we did not employ any third party libraries; all of the training

and classification code is implemented from scratch to make the code more amenable to future research involving Android development.

Our tool visualizes a home setting with multiple rooms and generates multiple lights in each room. It allows the simulations to be conducted in a 2D map of the home and offers the ability to interactively perform actions on the devices in the simulated home by moving an actor image around the map.

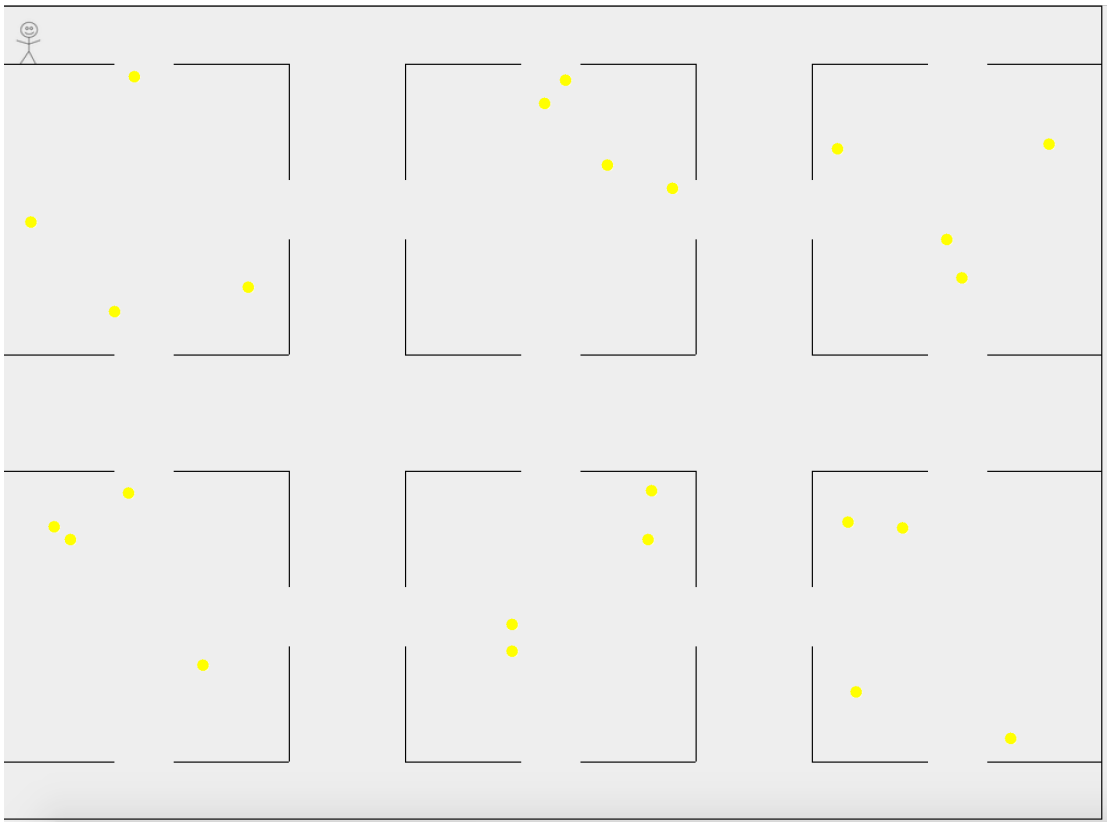

Figure 4.8: The first view of the map at the beginning of simulation

In the simulator, each room has its dimensions in the map with walls and doors, as well as a number of light devices available and an individual data table for the interaction

history. Each room constructs its own learning model since the dynamics and light settings of each room are different in the real world, e.g. kitchen lights are set and used in a different way than living room lights.

Besides the movements of the actor, there are two actions available in terms of device interactions: as “turn on lights” and “feedback”. By pressing the Enter key, the predicted device is turned on based on the actor’s current location and the actor’s history with the devices in that room. This Enter key represents “turn on light” button in an app that is implemented based on the proposed system for remote home control. If it is the first time for the actor to interact with this particular room, the closest light to the actor is turned on as a result of the first turn on action. As the new interactions form, the machine learning models becomes more effective, which will be discussed in the results and analysis sections.

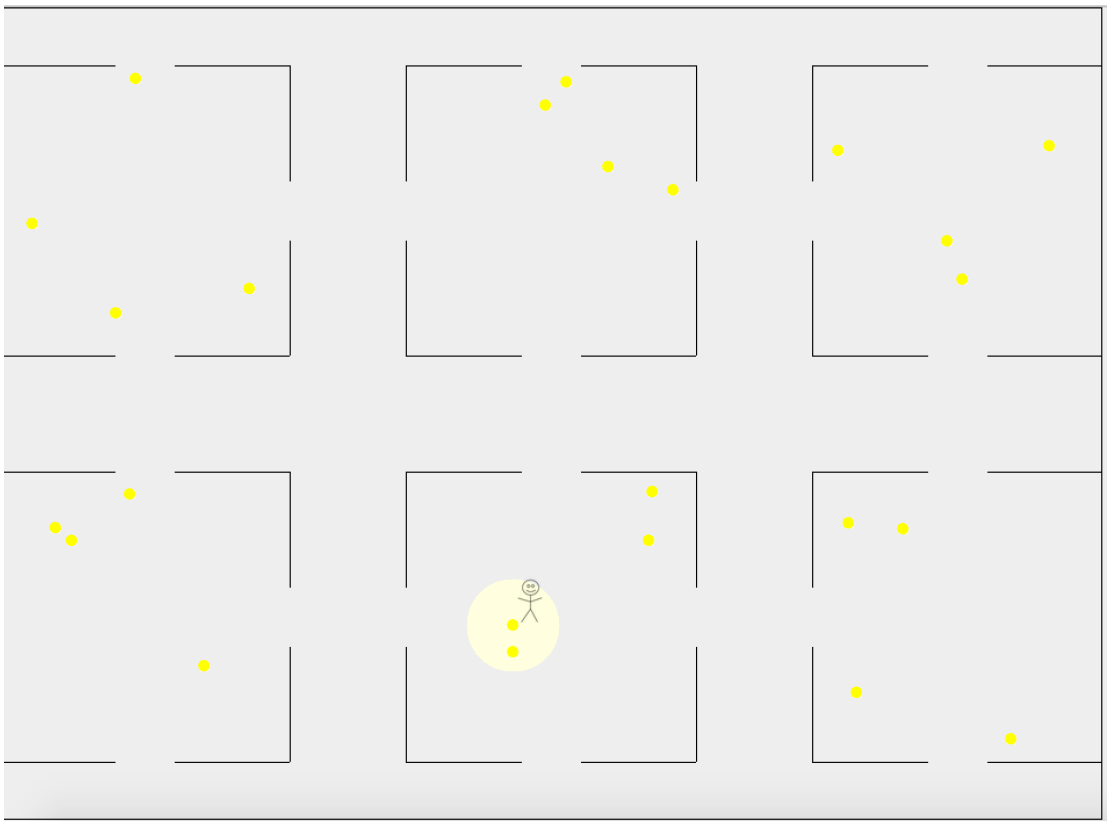

Figure 4.9: View after turning on one light

There is also a feedback mechanism for the situations when the user is not happy with the light turned on. When a light is turned on by the simulator and the user is not happy with this prediction, there is a ten seconds feedback time for the user to undo this turn on action before the system to accept it as the right device to turn on under these circumstances and save this interaction in the database. The duration of the feedback time is a parameter and can be easily changed according to preferences. The Space key is enabled for the user to give a negative feedback after a turn on action, representing "undo" button in an app implemented based on this system. If this feedback is within the

feedback time, the current light turned on is turned off by the system. This last prediction is perceived as a wrong one and will not be given as the decision to the user's next turn on action. Instead, the scoring order of the lights is exploited, which is provided by the learning algorithms. This scoring order is also used to turn on multiple lights in a sequence in a simulator environment. For example, if the light that has the best score from learning models is already turned on, the system responds to additional turn on actions in that room, as following the scoring order of lights.

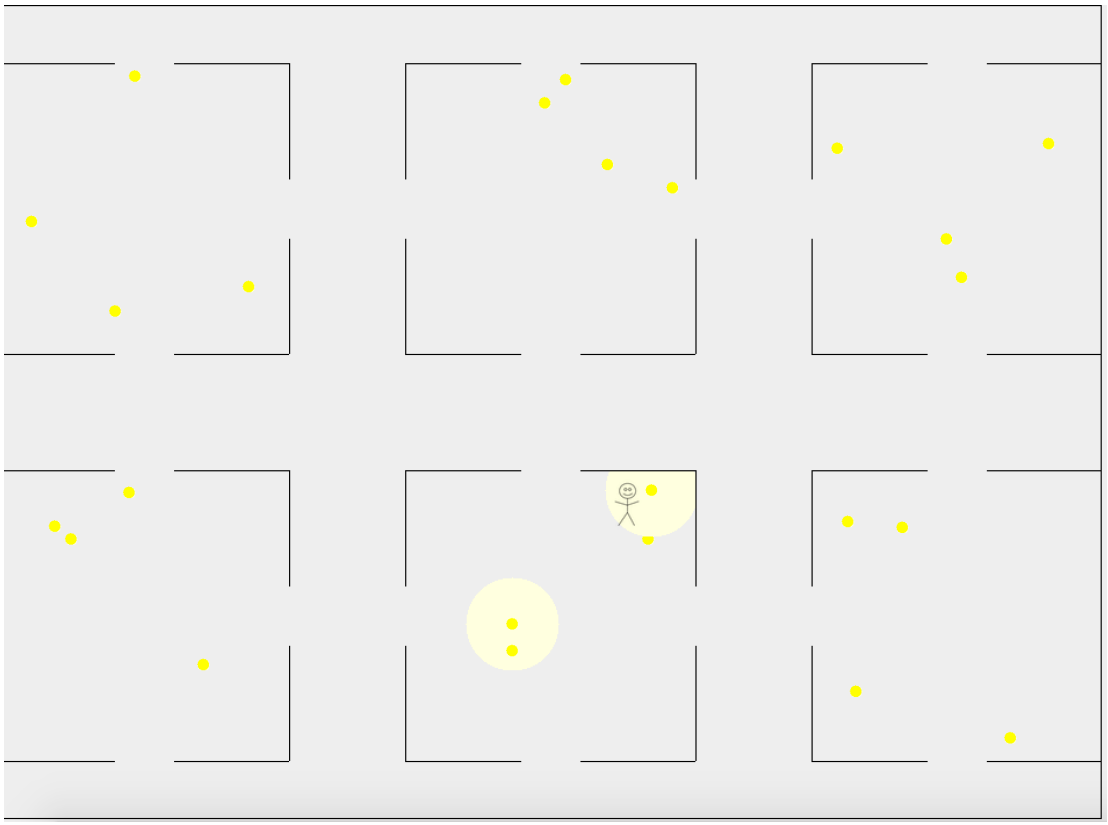

Figure 4.10: Turning on Multiple Lights in Simulator

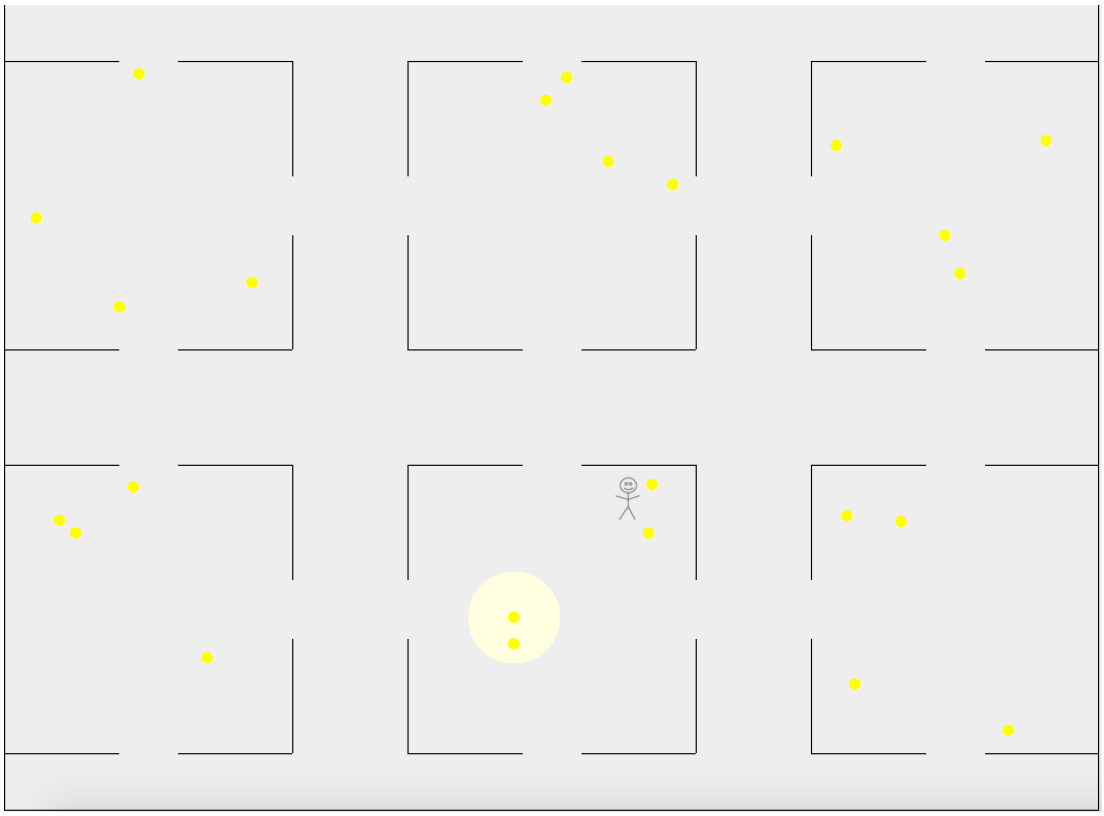

Figure 4.11 View after the User Feedback in Simulator

Chapter 5: Results and Analysis

In this section, we show the behavior of the model using different datasets and data sizes. Datasets are generated for different scenarios and the system is evaluated for both the Softmax Regression and K-Nearest Neighbor algorithms.

5.1. RESULTS WITH DATA FOR 20 DAYS

Twenty days of realistic data is generated for a person with a turn-on action pattern for a kitchen with four distinct lights defined with the following patterns:

- Light 1 is turned on between 7 am and 9 am, while the ambient light intensity is in the range of 50% and 70%.
- Light 2 is turned on between 11 am and 2 pm, while the ambient light intensity of the room is in the range of 70% and 95%.
- Light 3 is turned on between 5 pm and 9 pm, while the ambient light intensity of the room is in the range of 40% and 80%.
- Light 4 is turned on between 12 am and 5 am, while the ambient light intensity of the room is in the range of 1% and 40%.

This data represents the user interactions between 1st of June and 20th of June 2017. Each interaction is represented with a data point, adding up 80 data points for 20 days with four lights. The following parts talk about the results of each learning model for this dataset with different noise levels.

5.1.1 Results without noise

The figures below show the 10-fold cross validation results with this data for Softmax algorithms with both batch learning and online learning and, also KNN hyper parameter search results over the number of neighbors (k). For the accuracy comparison and confusion matrix, the best performing neighbor configuration is chosen for KNN.

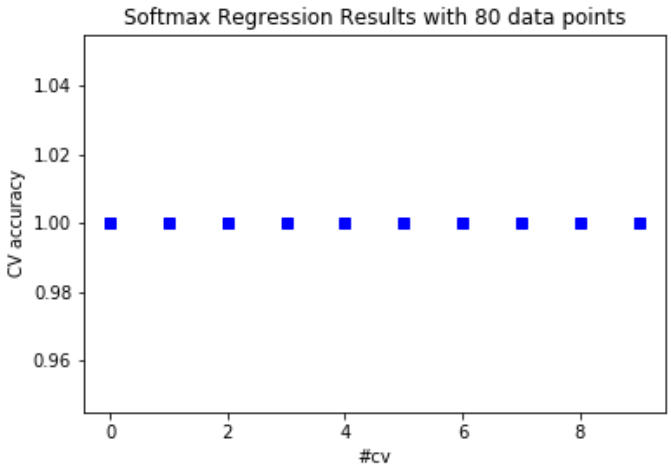

Figure 5.1: Softmax Regression Results with Batch Learning

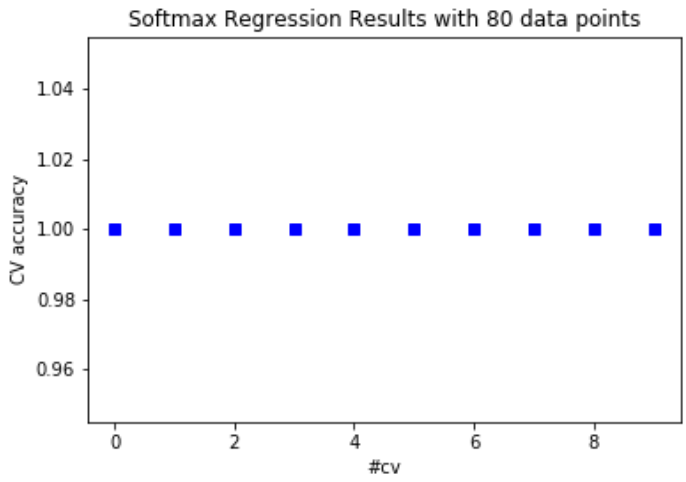

Figure 5.2: Softmax Regression Results with Online Learning

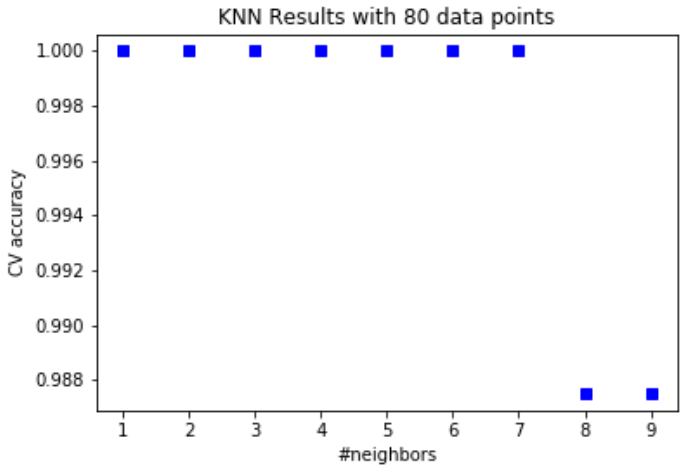

Figure 5.3: KNN Results over the Number of Neighbors

As seen in the figures, the mean accuracy scores for both KNN and Softmax Regression for both batch learning and online learning are 100%, which demonstrates

that even if the data size is as small as 80 points, both Softmax Regression and KNN algorithms are able to perfectly capture the user interaction pattern in the case where the lights are turned on at distinct times separately.

Moreover, the confusion matrices are constructed to investigate the performance in detail for each light.

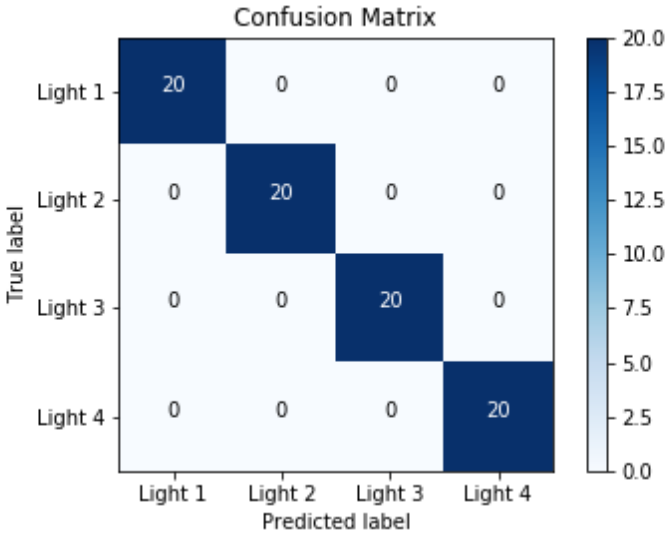

Figure 5.4: Confusion Matrix of Softmax Regression with Batch Learning

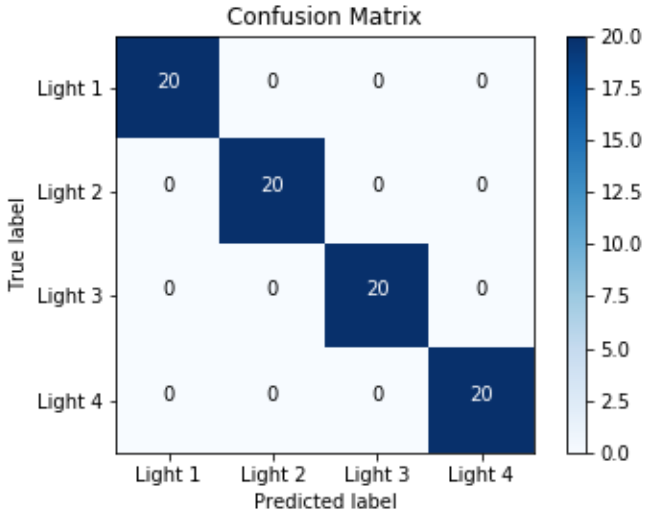

Figure 5.5: Confusion Matrix of Softmax Regression with Online Learning

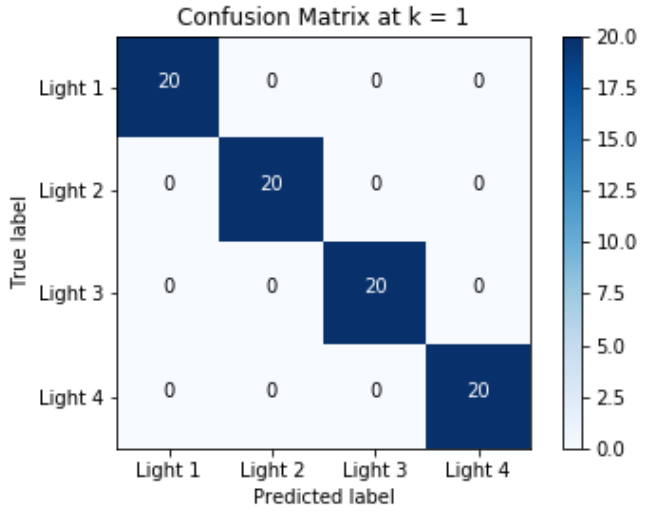

Figure 5.6: Confusion Matrix of KNN with the Best Scored Neighbor Configuration

As seen in the plots and the confusion matrices, all algorithms work perfectly with this dataset where the time slot for each light is different than the other. We call this dataset the perfect situation since there is no changing behavior and the time slot for each light is completely separated.

5.1.2 Results with 10% noise

In order to test the system performance under unexpected interaction behaviors, we added 10% noise to the same dataset, where this noise is generated with totally random values for times in the range of 24 hours, light intensities in the range of 1% – 100% with random light ids in the range of 1 to 4. 10-fold cross validation results for Softmax Regression and KNN hyper parameter search results over the number of neighbors (k) can be found in the following Figures.

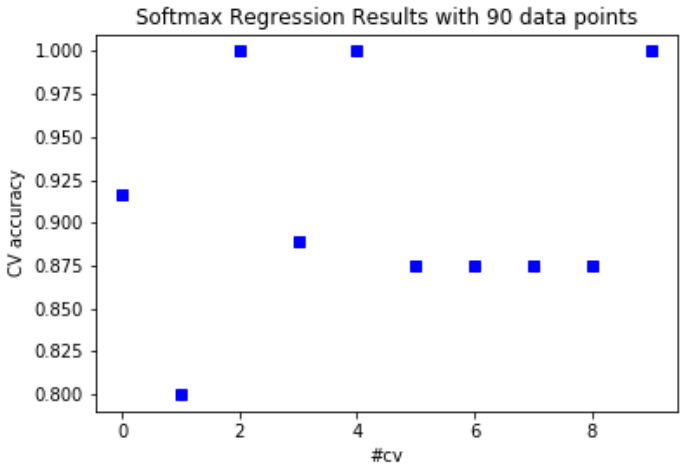

Figure 5.7: Softmax Regression Results with 10% Noise for Batch Learning

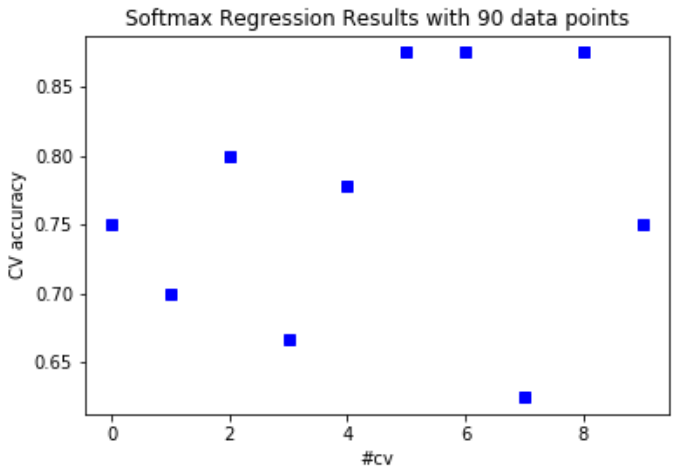

Figure 5.8: Softmax Regression Results with 10% Noise for Online Learning

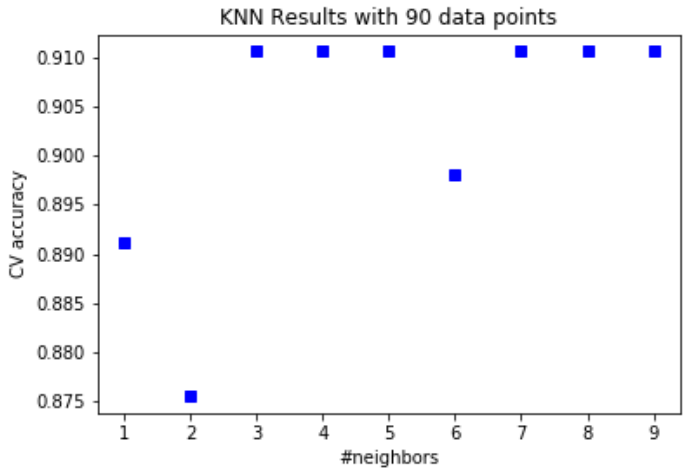

Figure 5.9: KNN Results with 10% Noise over the Number of Neighbors

In the case with 10% noise, the mean accuracy for the Softmax Regression algorithm with batch learning is 91% while that with online learning is 77%. Moreover, KNN also resulted in 91% in terms of mean accuracy.

Even if the randomly generated data points, which represent the unexpected fluctuations in the pattern, result in a decrease in the accuracy scores, the performance of both Softmax Regression with batch learning and KNN are still more than 90% while the online learning performance is just slightly behind them.

The confusion matrices for each algorithm can be found in the following figures.

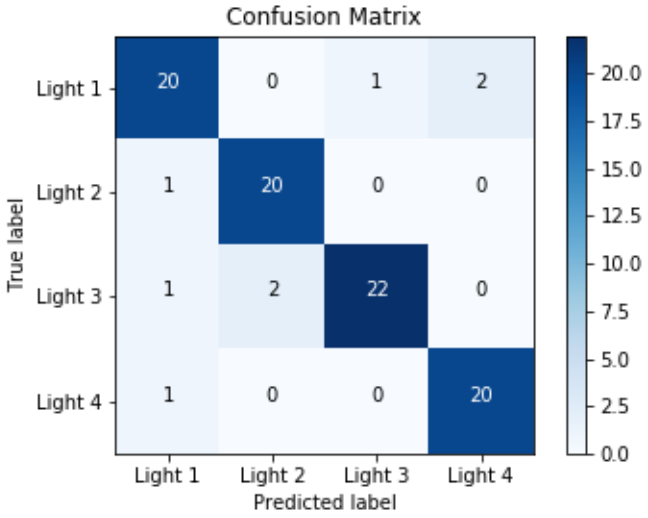

Figure 5.10: Confusion Matrix of Softmax Regression with Batch Learning

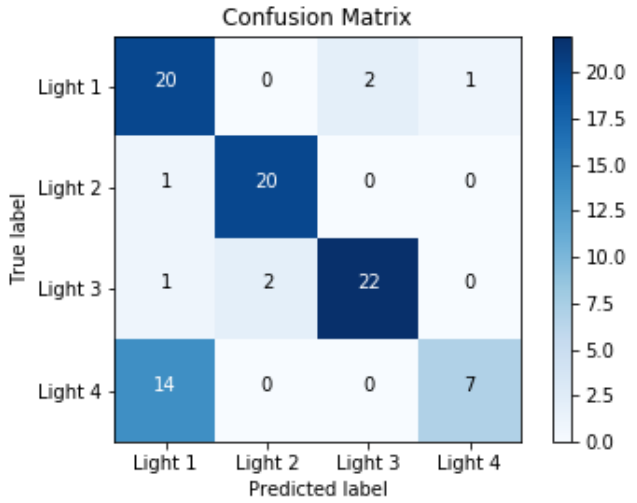

Figure 5.11: Confusion matrix for Softmax Regression with Online Learning

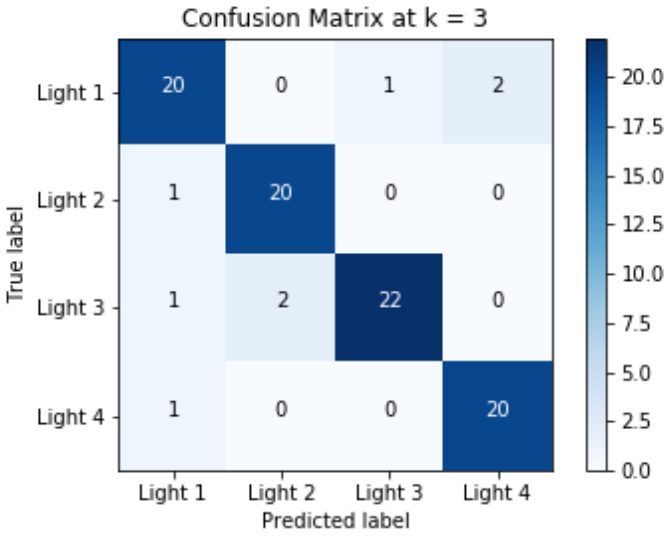

Figure 5.12: Confusion Matrix of KNN with the Best Neighbor Configuration

5.1.3 Results with 20% noise

This section shows the results we we added 20% noise, where the number of randomized data points are increased in the dataset in order to resemble the unexpected user behaviors. As in the previous sections, 10-fold cross validation is done for Softmax Regression for both batch learning and online learning methods, and KNN hyper parameter search is performed over the number of neighbors (k).

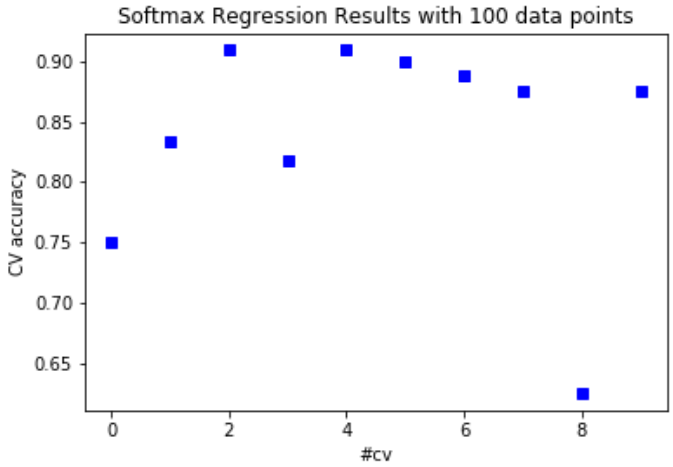

Figure 5.13: Softmax Regression Results with 20% Noise for Batch Learning

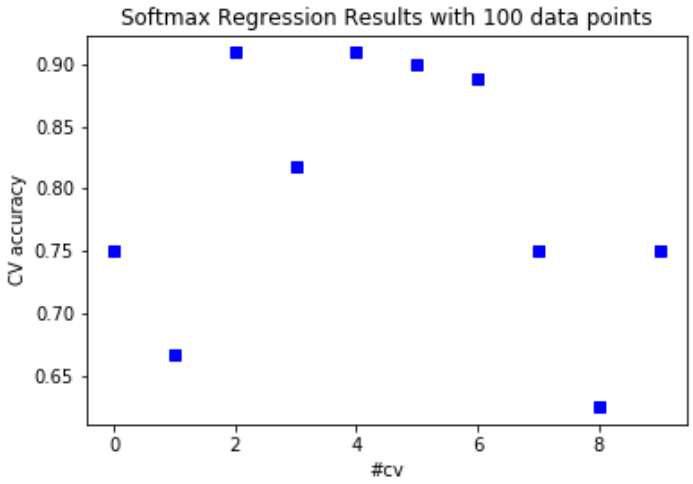

Figure 5.14: Softmax Regression Results with 20% Noise for Online Learning

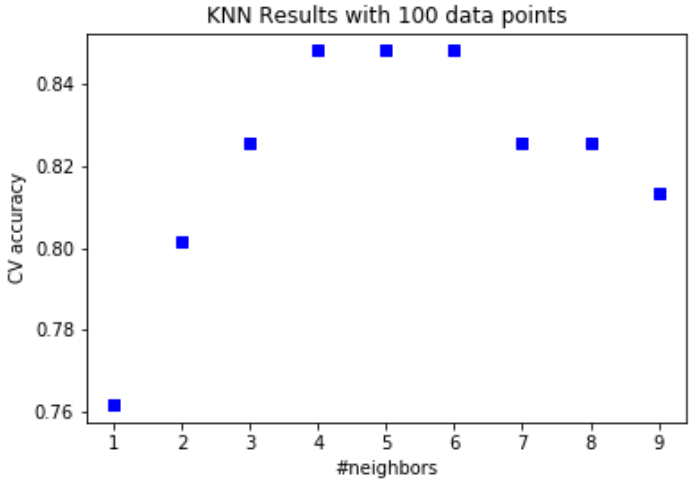

Figure 5.15: KNN Results over the Number of Neighbors with 20% Noise

As seen in the figures above, the mean accuracy for Softmax Regression with online learning is 80% and that with batch learning is 84%, while the mean accuracy for KNN with the best scored neighbor configuration is 85%.

The confusion matrices for each algorithm can be found in the following figures.

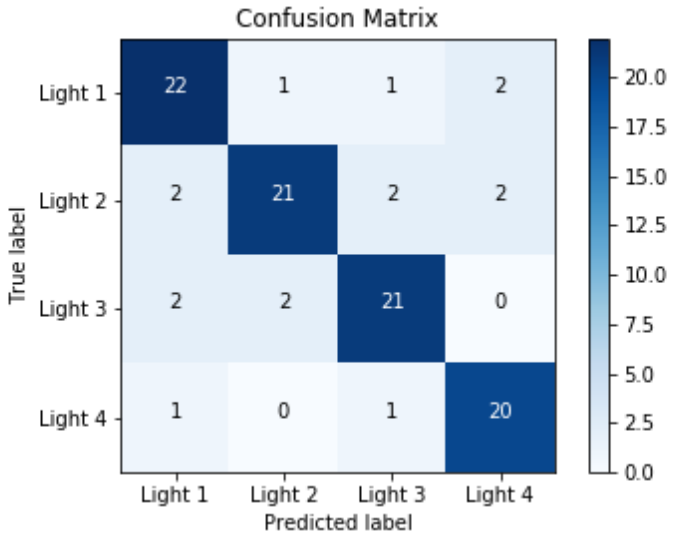

Figure 5.16: Confusion Matrix of Softmax Regression with Batch Learning

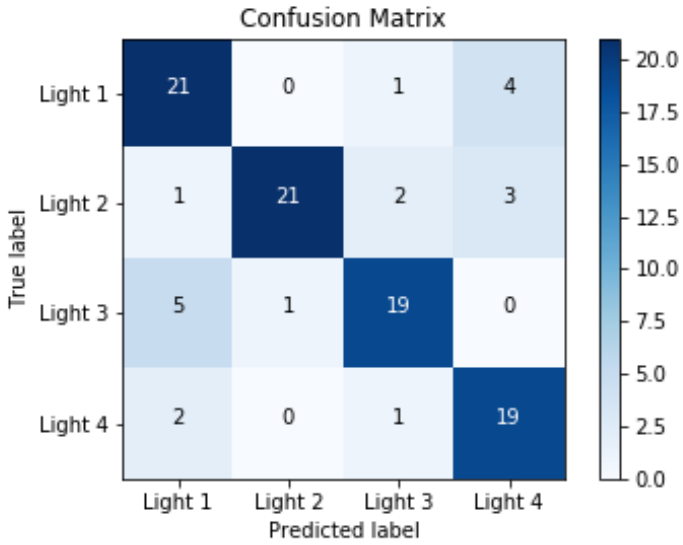

Figure 5.17: Confusion Matrix of Softmax Regression with Online Learning

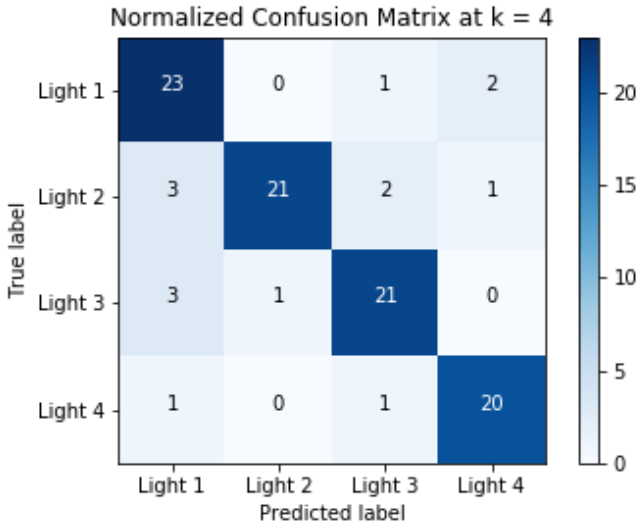

Figure 5.18: Confusion Matrix of KNN with the Best Performed Neighbor Configuration

5.2 Effect of the Light Intensity Reading on Results

We also generated another dataset to validate the effect of the light intensity reading on the accuracy. This dataset consists of two lights in such a way that the user turns on each of the lights in the same time slot in the same season but with different light intensities of the room. Basically, every feature is set fixed in the dataset except the light intensity reading. The details about the dataset are:

- Light 1 is turned on between 4 pm to 5 pm in June 2017 when the ambient light intensity is in the range of 30% and 50%.
- Light 2 is turned on between 4 pm to 5 pm in June 2017 when the ambient light intensity is in the range of 60% and 80%.

This scenario is also important for the cases when the user turns on multiple lights at the same time slot in a sequential way. When the first light is turned on, the light intensity of the room increases, and this difference in the light intensity is the determining factor for the subsequent turn on actions leading to capture the sequential pattern and turn on correct multiple lights in the same time slot.

We generated 50 data points for each light respecting this pattern. The following figures present 10-fold cross validation results of Softmax Regression with batch learning and online learning, and KNN hyper parameter search over the number of neighbors (k) for this dataset.

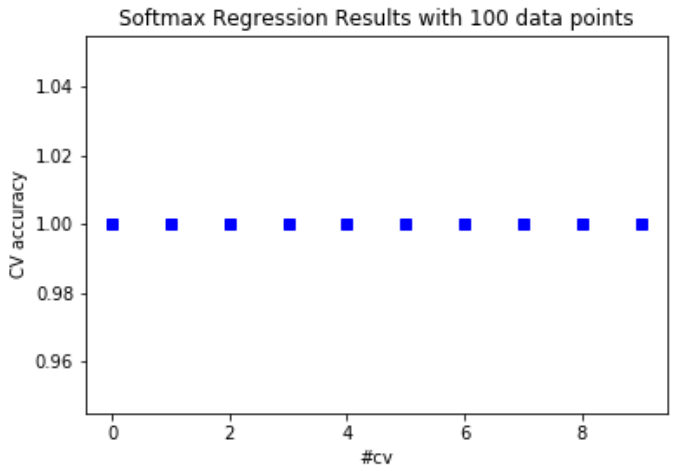

Figure 5.19: Softmax Regression Results with Batch Learning for Light Intensity Effect

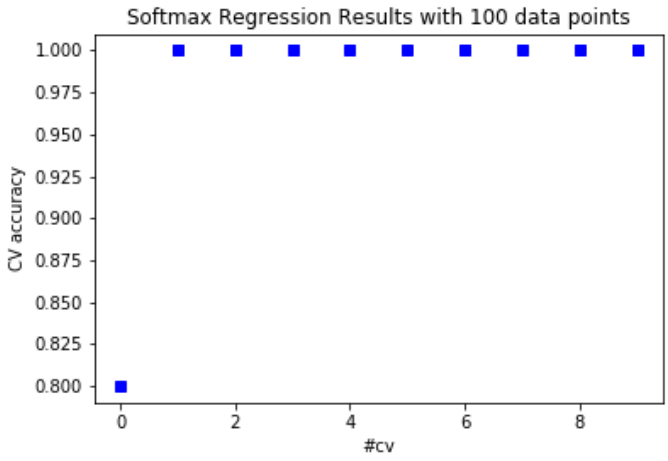

Figure 5.20: Softmax Regression Results with Online Learning for Light Intensity Effect

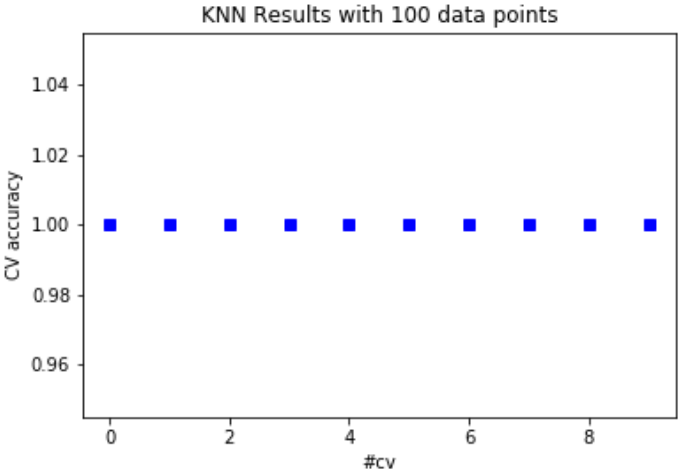

Figure 5.21: KNN results over the Number of Neighbors for Light Intensity Effect

As seen in the figures above, both Softmax Regression with batch learning and KNN performed 100% accurately for capturing the user pattern when the light intensity is the only determining factor while the accuracy of Softmax Regression with online learning for this case is 98%, which is still pretty good.

The confusion matrices for both Softmax Regression algorithms and KNN can be found in the following figures.

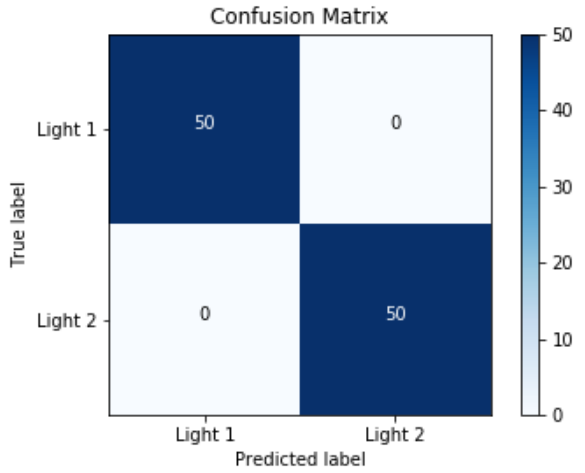

Figure 5.22: Confusion Matrix of Softmax Regression with Batch Learning

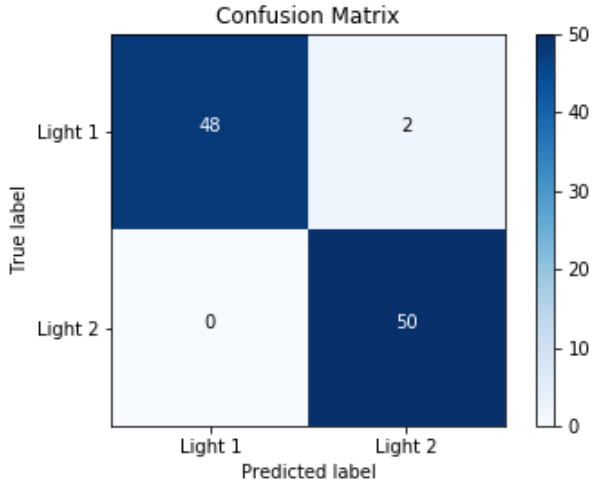

Figure 5.23: Confusion Matrix of Softmax Regression with Online Learning

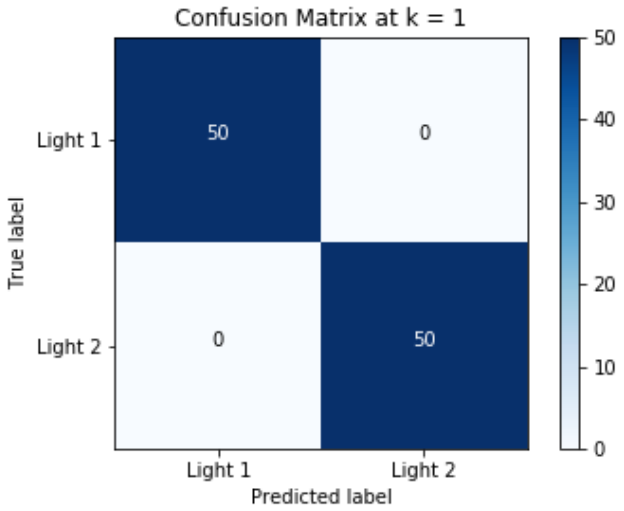

Figure 5.24: Confusion Matrix of KNN with the Best Scored Neighbor Configuration

As seen in the both plots and confusion matrices, all three methods are able to accurately distinguish the behavior pattern for the case when the time slots of the interactions with these two lights are the same, and the light intensity is the only determining factor. This ability of the models leads the system to capture the user's unique sequence of turning on actions for multiple lights in the room.

5.3 RESULTS FOR REALISTIC DATASETS WITH DIFFERENT DATA SIZES

This part of the paper is dedicated for the behavior of the system according to the number of data for a realistic dataset. The dataset is generated with the date range of 365 days, so there is a variety of seasons covered. There is also a behavior change with respect to the seasons reflected in the dataset. The details of the dataset scenario are:

For the dates between 21th March 2017 and 23th September 2017:

- Light 1 is turned on between 7 am and 10 am when the ambient light intensity is in the range of 60% and 80%.
- Light 2 is turned on between 12 pm and 3 pm when the light ambient intensity is in the range of 60% and 90%.
- Light 3 is turned on between 3 pm and 8 pm when the light ambient intensity is in the range of 60% and 90%.
- Light 4 is turned on between 12 am and 4 am when the ambient light intensity is in the range of 1% and 20%.
- Light 3 is rarely turned on between 5 am and 7 am when the ambient light intensity is in the range of 1% and 40%.

For the dates between 23th September 2016 and 21th March 2017:

- Light 1 is turned on between 7 am and 10 am when the ambient light intensity is in the range of 20% and 70%.
- Light 2 is turned on between 12 pm and 3 pm when the ambient light intensity is in the range of 50% and 80%.
- Light 3 is turned on between 3 pm and 6 pm when the ambient light intensity is in the range of 40% and 60%.
- Light 4 is turned on between 7 pm and 2 am when the ambient light intensity is in the range of 1% and 30%.

- Light 3 is rarely turned on between 2 am and 7 am when the ambient light intensity is in the range of 1% and 30%.

The following sections describe the results of the same realistic scenario with different dataset sizes.

5.3.1. Results with small data size

The size of the dataset is built small in order to test the limits of the models to adapt themselves to the user interactions in the case of the data less than the number of days covered. This dataset consists of 180 points. The following figures demonstrate 10-fold cross validation results of Softmax Regression with batch learning and online learning, and KNN hyper parameter search over the number of neighbors (k) for this dataset.

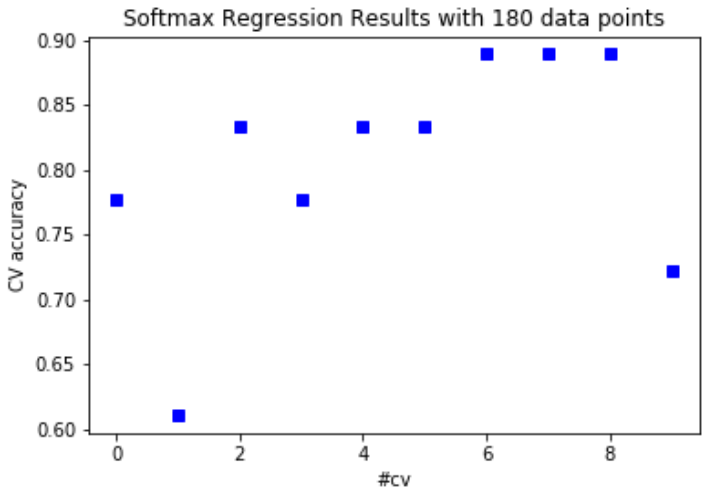

Figure 5.25: Softmax Regression Results with Batch Learning

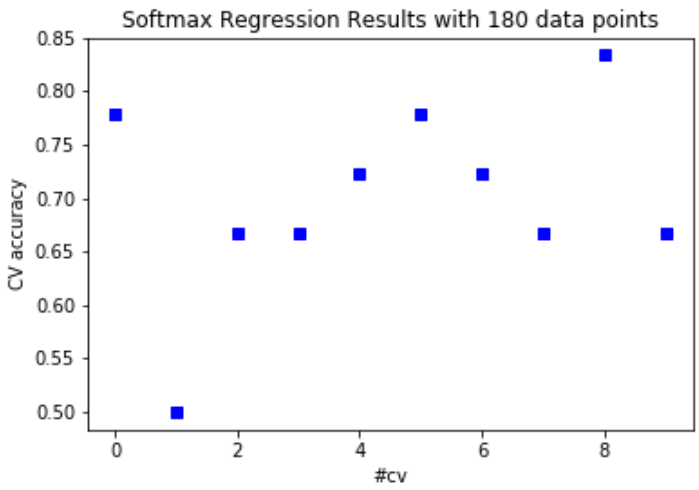

Figure 5.26: Softmax Regression Results with Online Learning

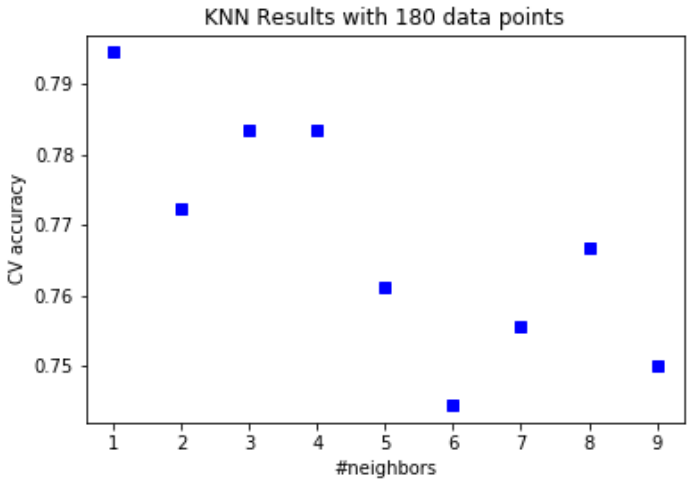

Figure 5.27: KNN Results over the Number of Neighbors

The mean accuracy result for Softmax Regression with batch learning is 81% in this case while that of online learning is 74%. Moreover, KNN performed with 80% accuracy. The confusion matrices can be found in the following figures.

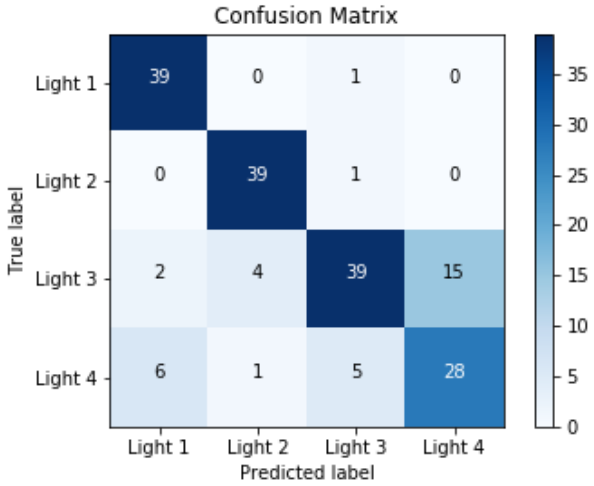

Figure 5.28: Confusion Matrix of Softmax Regression with Batch Learning

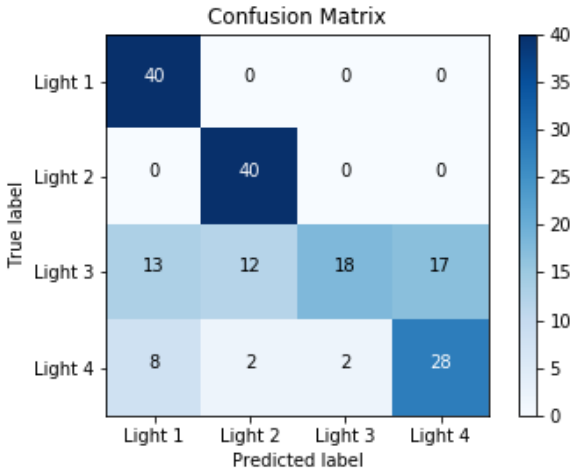

Figure 5.29: Confusion Matrix for Softmax Regression with Online Learning

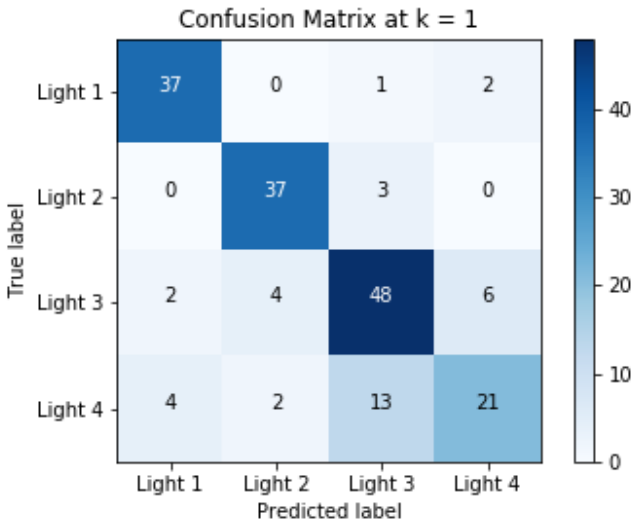

Figure 5.30: Confusion Matrix of KNN with the Best Performed Neighbor Configuration

At this moment, it would be good to take the data size into account once more. The number of data points is 180 in total but different seasons are covered with their different interaction behaviors for all the year, resulting approximately only one interaction in the room per two days. Still, the usage patterns of light 1 and light 2 are well captured by all three learning methods as seen in the confusion matrices, which can be reasoned in such a way that lights 1 and 2 are used somewhat more consistently throughout the year compared to lights 3 and 4. Lights 3 and 4 are used interchangeably at some time slots depending on the seasons, and they have more diverse ranges of interaction patterns throughout the year. From the confusion matrix it seems that this data size is not sufficient for the online learning algorithm to capture the usage pattern changes in different seasons for light 3. Moreover, even if all learning methods are

somewhat able to distinguish the correct situations for turning on light 4, there is still some space for improvement by expanding the data size.

5.3.2 Results with medium data size

The same scenario is also built with 900 data points, which is a more reasonable amount of interactions for a year compared to the previous data set. The following figures demonstrate 10-fold cross validation results of Softmax Regression with batch learning and online learning, and KNN hyper parameter search over the number of neighbors (k) for this dataset.

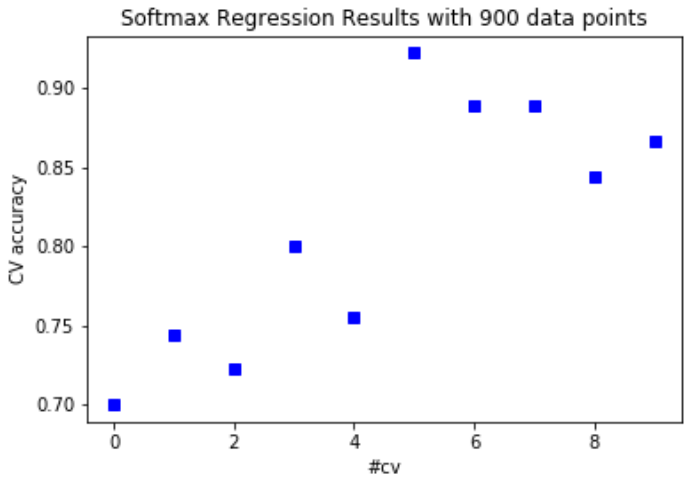

Figure 5.31: Softmax Regression Results with Batch Learning

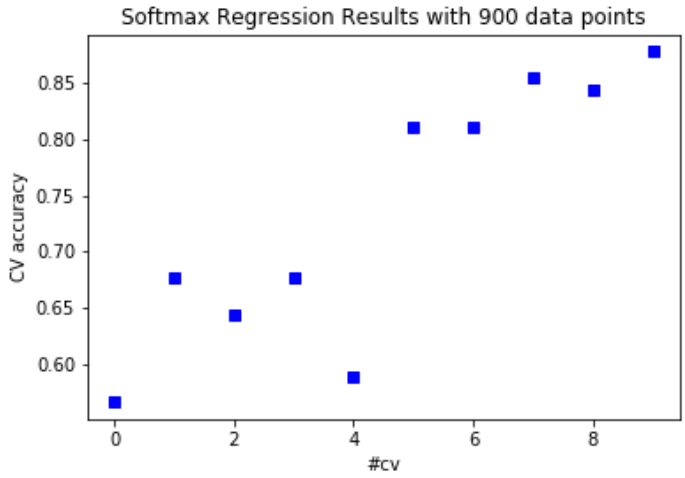

Figure 5.32: Softmax Regression Results with Online Learning

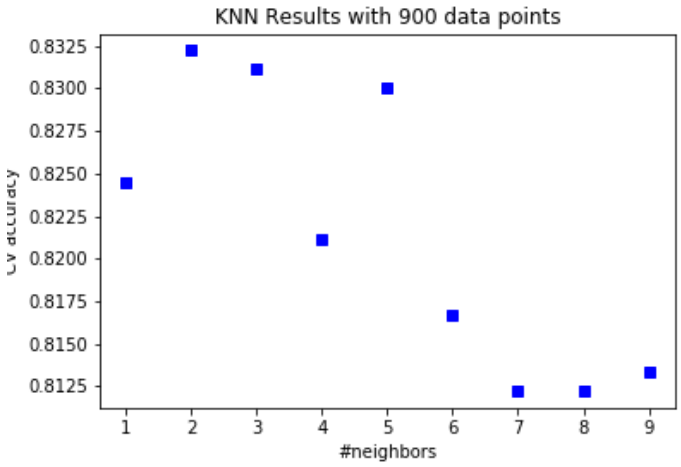

Figure 5.33: KNN Results over the Number of Neighbors

The mean accuracy for the Softmax Regression algorithm with batch learning is 81% and that of KNN is 83% for the best performed neighbor configuration while Softmax Regression with online learning performed with 74% mean accuracy.

The confusion matrices for all three learning methods can be found in the following figures.

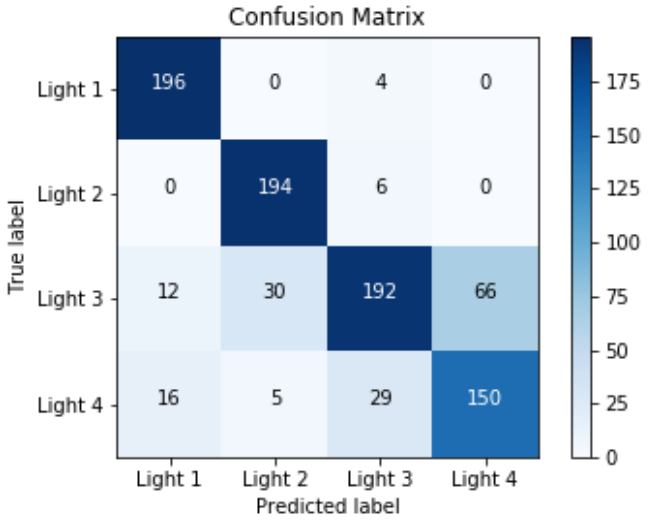

Figure 5.34: Confusion Matrix of Softmax Regression with Batch Learning

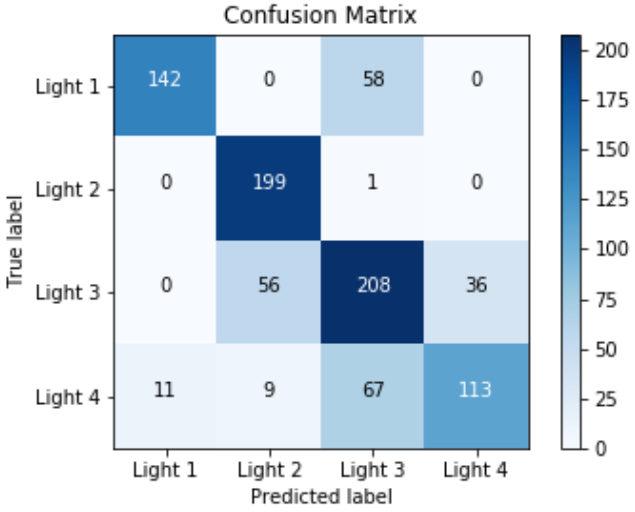

Figure 5.35: Confusion Matrix of Softmax Regression with Online Learning

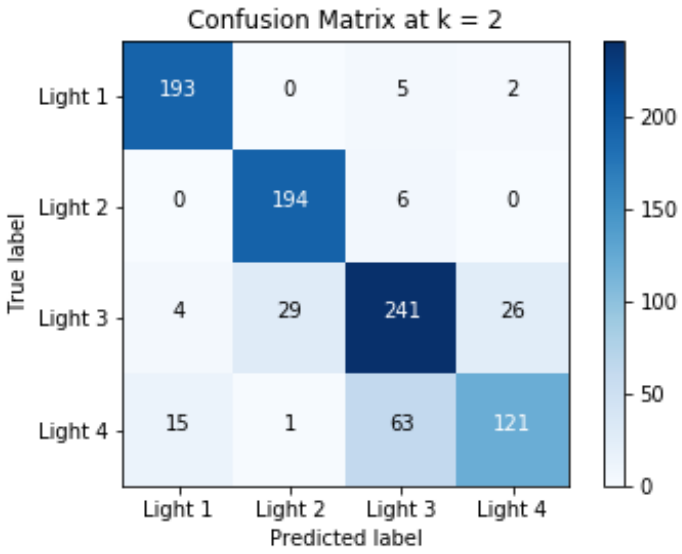

Figure 5.36: Confusion Matrix of KNN with the Best Performed Neighbor Configuration

There are some slight improvements in terms of overall accuracy for KNN and the Softmax Regression with batch learning. While the overall accuracy for the Softmax Regression seems to stay the same, the true positives for light 3 improve compared to the previous case with the small data size. However, in the online learning, there is a decay in the ability to distinguish the pattern of light 1 from other lights. As seen in the confusion matrix of the online learning, light 1 is confused with light 3. The reason may be that the number of data points of interactions with light 3 is higher than that of light 1, and there is a time interval shared between light 1 and light 3. Overall, the online learning is still open to improvement by the increasing the data size in order to reduce the effect of these overlaps, considering KNN and Softmax Regression with batch learning performs better than 80% so far.

5.3.3 Results with big data size

A dataset with the same scenario is constructed with the number of data points as 8500. This dataset represents the case when this particular room is really popular and takes more than 20 light device interactions in a day. The following figures demonstrate 10-fold cross validation results of Softmax Regression with batch learning and online learning, and KNN hyper parameter search over the number of neighbors (k) for this dataset.

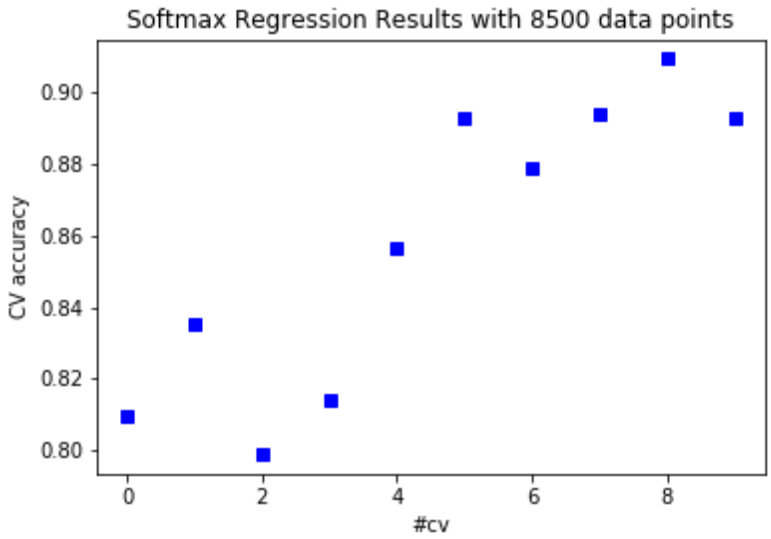

Figure 5.37: Softmax Regression Results with Batch Learning

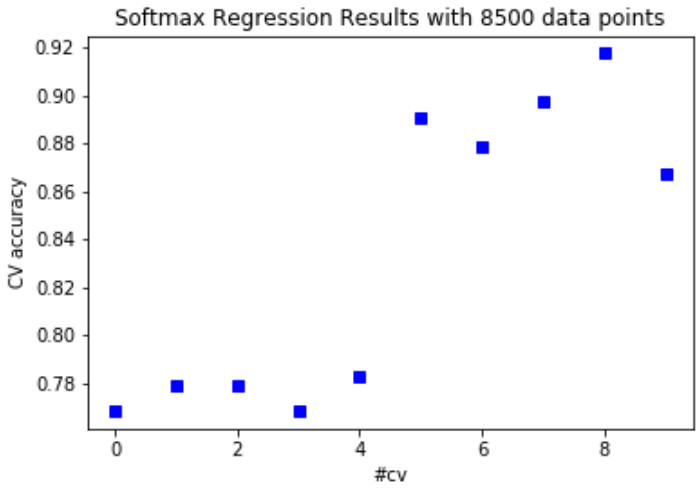

Figure 5.38: Softmax Regression Results with Online Learning

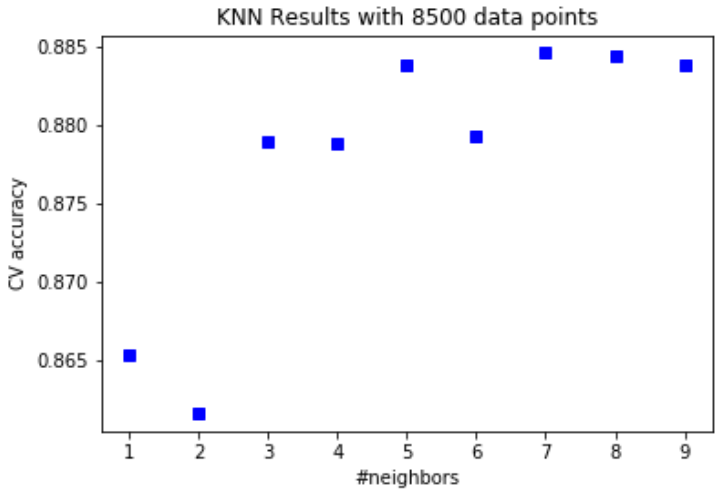

Figure 5.39: KNN Results over the Number of Neighbors

The mean accuracy for Softmax Regression with batch learning is 86% while that of KNN with the best performed neighbor configuration is 89%. Moreover, Softmax

Regression with the online learning method performed with a mean accuracy of 84%, which is a significant improvement compared to the results with previous datasets, which were 74% accuracy.

The confusion matrices for all three learning methods can be found in the following figures.

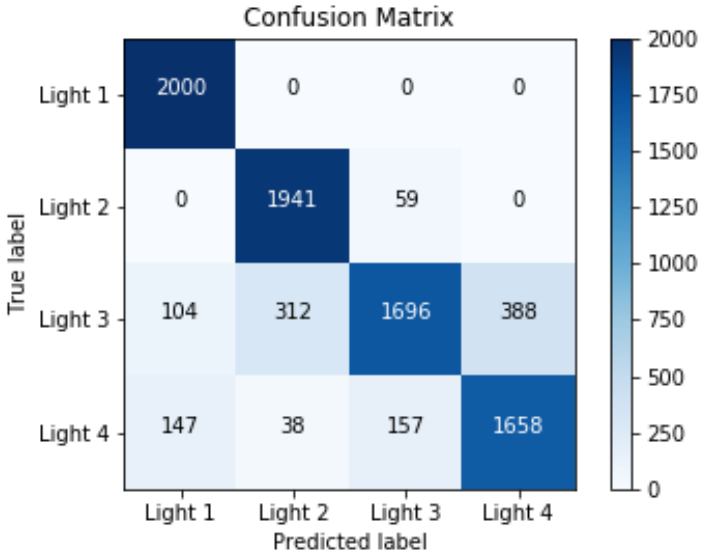

Figure 5.40: Confusion Matrix of Softmax Regression with Batch Learning

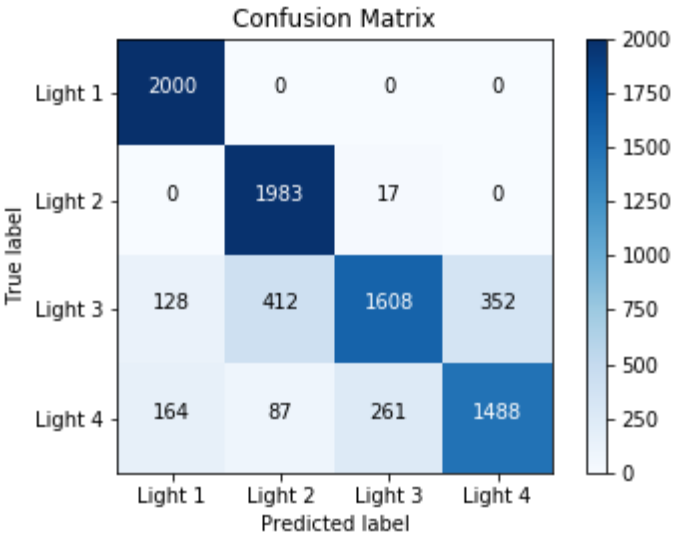

Figure 5.41: Confusion Matrix of Softmax Regression with Online Learning

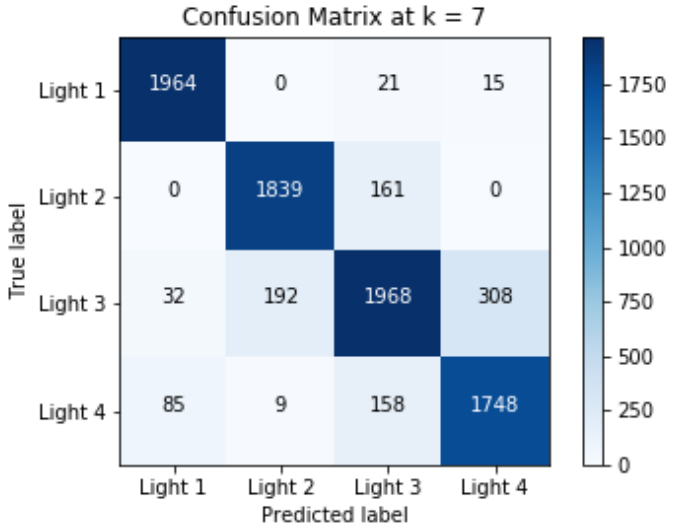

Figure 5.42: Confusion Matrix of KNN with the Best Scored Neighbor Configuration

As the data size becomes larger, the accuracy of the online learning algorithm becomes closer to the performance of the batch learning and KNN. From the confusion matrix of the online learning method, it can be seen that the online learning method is adapted to the user's interaction patterns with the lights. Also, both Softmax Regression with batch learning and KNN perform well with the increased number of data samples as expected.

	Batch SR	Online SR	KNN
180 data samples	81%	74%	80%
900 data samples	81%	74%	83%
8500 data samples	86%	84%	89%

Table 1: Accuracy Results Comparison Table for Clean Realistic Dataset

As can be seen from the Table above, the KNN algorithm outperformed the other algorithms in overall. However, it should be considered that KNN requires storing all the training data samples and going over this data every time when making the prediction so that when the data size becomes really large, the space and time complexity will be huge. Similarly, Softmax Regression with batch learning requires offline training data and replaces its parameters based on training on this data. Since the model is taking into account the whole dataset, as in KNN, training may take a lot of time when the dataset is really large while prediction with the batch learning still happens faster than KNN.

Therefore, both KNN and Softmax Regression with batch learning result in better performance in terms of accuracy at the expense of time and space for training. However, the online learning method proposed in this work trains itself one data point at a time as the new data comes, and changes are directly reflected in the weights and the bias term eliminating the need for storage of past interaction history. As opposed to both offline algorithms, KNN and batch Softmax Regression, described before, this online learning method is faster since it only uses one data point to calculate the gradient and update its parameters accordingly. Still, it can be concluded that even in the case of a complex dataset, the online learning algorithm offers comparable results when there is enough data.

Chapter 6: Conclusions and Future Work

This thesis proposed three learning model approaches based on device interaction history of the user in order to personalize decision making of human-centered smart lighting environments. These three learning methods, namely K-Nearest Neighbor, Softmax Regression with batch learning and Softmax Regression with online learning, capture the characteristics of an individual's interactions with smart devices and any changes in the behavior and adapt themselves accordingly. Moreover, the thesis introduces a Java simulator, which interactively demonstrates the behavior of a personalized smart home setting with the integrated learning models. The user of the simulator has a variety of capabilities such as directing the actor in the house, interacting with the light devices in different rooms as well as giving feedback to the decision making mechanism of the system in case the system turns on an undesired light. The simulator leads up to a way of evaluating adaptive learning models in smart home environments, reducing the immediate need of testing their behavior in a real life smart home, which is both hard and costly to construct and maintain.

The learning models are evaluated with different datasets, as explained in the Results and Analysis chapter. All three algorithms work perfectly in situations when interactions with distinct "things", smart light devices, are separable in terms of their corresponding context. For more complicated datasets, which have overlaps between the usage context of devices and big changes in user behavior over time, the results show that

the online learning algorithm needs more data in order to catch the performance of KNN and SR batch learning algorithms. Of course, this is not surprising considering the fact that in online learning the model incrementally trains itself with one data point at a time. On the other hand, as discussed before at the end of Results and Analysis chapter, the presented online learning method is the fastest and requires the least space among all three proposed models. In order to address these tradeoffs, in future applications following this work, the implementation can be done in such a way that at the beginning, predictions are prepared based on KNN while also training SR, and after there is enough data, the decision making mechanism switches to the online learning method, eliminating the need of storing interactions data continuously. Furthermore, as another future work, in order to obtain enough relevant data faster in time, the data points of family members can be shared and used across them since the family may have similar interaction patterns with the devices at home.

One thing that is not addressed in this thesis work is the privacy issues of storing user-device interactions in the system. Both KNN and SR with batch learning require the whole past device interaction history of the user for training, while SR with online learning just needs the latest interaction for incremental training. This privacy aspect should be also taken into account for the implementations of real life applications following this work.

Bibliography

[1] https://en.wikipedia.org/wiki/Internet_of_things

[2] Diana J. Cook, Aaron S. Crandall, and Brian L. Thomas. CASAS: A smart home in a box. In *Computer, vol. 46 no. 7*, pages 62-69. IEEE, 2013.

[3] Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy. ARAS human activity datasets in multiple homes with multiple residents. *Proceedings of 7th International Conference on Pervasive Computing Technologies for Healthcare*, pages 232-235. ACM, 2013.

[4] Arni Ariani, Stephen J. Redmond, David Chang, and Nigel H. Lovell. Simulation of a smart home environment. *Proceedings of the 2013 3rd International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering (ICICI-BME)*, pages 27-32. IEEE, 2013.

[5] Kevin Bouchard, Amir Ajroud, Bruno Bouchard, and Abdenour Bouzouane. SIMACT: A 3D open source smart home simulator for activity recognition. *Lecture Notes in Computer Science, vol 6059*, pages 524-533. Springer-Verlag, 2010.

[6] Shiu Kumar. Ubiquitous smart home system using Android application. *International Journal of Computer Networks & Communications (IJCNC), vol. 6*, pages 33-43, 2014.

[7] Rajeev Piyare. Ubiquitous home control and monitoring system using Android based smart phone. *International Journal of Things, vol. 2*, pages 5-11, 2013.

[8] Malik Sikandar Hayat Khiyal, Aihab Khan, and Erum Shehzadi. SMS based wireless home appliance control system (HACS) for automating appliances and security. *Issues in Informing Science and Information Technology, vol. 6*, pages 887-894, 2009.

[9] https://www2.meethue.com/en-us/light-your-home-smarter/daddys-home-light

[10] https://www.wink.com/products/wink-bright-smart-lighting-essentials/

[11] Dimitris Lymberopoulos, Jie Liu, Xue Yang, Romit Roy Choudhury, Vlado Handziski, and Souvik Sen. A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned. *Proceedings of 14th International Conference on Information Processing in Sensor Networks*, pages 178-189. IEEE/ACM IPSN, 2015.

[12] Xiao Jiang, Zimu Zhou, Youwen Yi, and Lionel M. Ni. A survey on wireless indoor localization from the device perspective. *ACM Computer Surveys (CSUR) vol. 49*, pages 25:1-25:31. ACM, 2016.

[13] http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/