MODELING AND CONTROL TECHNIQUES IN SMART SYSTEMS

A Dissertation

by

LIJIA SUN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jiang Hu
Co-Chair of Committee,	Dana O. Porter
Committee Members,	Tie Liu
	Peng Li
Head of Department,	Miroslav M. Begovic

August 2018

Major Subject: Computer Engineering

ABSTRACT

Energy and food crisis are two major problems that our human society has to face in the 21st century. With the world's population reaching 7.62 billion as of May 2018, both electric power and agricultural industries turn to technological innovations for solutions to keep up the increasing demand. In the past and currently, utility companies rely on rule of thumb to estimate power consumption. However, inaccurate predictions often result in over production, and much energy is wasted. On the other hand, traditional periodic and threshold based irrigation practices have also been proven outdated. This problem is further compounded by recent years' frequent droughts across the globe. New technologies are needed to manage irrigations more efficiently.

Fortunately, with the unprecedented development of Artificial Intelligence (AI), wireless communication, and ubiquitous computing technologies, high degree of information integration and automation are steadily becoming reality. More smart metering devices are installed today than ever before, enabling fast and massive data collection. Patterns and trends can be more accurately predicted using machine learning techniques. Based on the results, utility companies can schedule production more efficiently, not only enhancing their profitabilities, but also making our world's energy supply more sustainable. In addition, predictions can serve as references to detect anomalous activities like power theft and cyber attacks.

On the other hand, with wireless communication, real-time soil moisture sensor readings and weather forecasts can be collected for precision irrigation. Smaller but more powerful controllers provide perfect platforms for complicated control algorithms. We designed and built a fully automated irrigation system at Bushland, Texas. It is designed to operate without any human intervention. Workers can program, move, and monitor multiple irrigation systems remotely. The algorithm that runs on the controls deserves more attention. AI and other state of art controlling techniques are implemented, making it much more powerful than any existing systems. By integrating professional crop yield simulation models like DSSAT, computers can run tens of thousand simulations on all kinds of weather and soil conditions, and more importantly, learn from the ex-

perience. In reality, such process would take thousands of years to obtain. Yet, the computers can find an optimum solution in minutes. The experience is then summarized as a policy and stored inside the controller as a lookup table. Furthermore, after each crop season, users can calibrate and update current policy with real harvest data.

Crop yield models like DSSAT and AquaCrop play very important roles in agricultural research. They represent our best knowledge in plant biology and can be very accurate when well calibrated. However, the calibration process itself is often time consuming, thus limiting the scale and speed of using these models. We made efforts to combine different models to produce a single accurate prediction using machine learning techniques. The process does not require manual calibration, but only soil, historical weather, and harvest data. 20 models were built, and their results were evaluated and compared. With high accuracy, machine learning techniques have shown a promising direction to best utilize professional models, and demonstrated great potential for use in future agricultural research.

DEDICATION

To my grandfather, who raised me up with care and love.

To my parents, who have firmly supported me in times good and bad.

To my uncle, who guided me through confusion and depression.

To the girl, whom I love and owed so dearly.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to Prof. Jiang Hu, my supervisor, who not only guided me in directions of research and study, but more importantly, taught and demonstrated the leadership and dedication to career. A special gratitude I give to Prof. Dana Porter, whose optimism and warm heart always inspired me at times of difficulty, and from whom I have received tremendous help. Furthermore, I would also like to thank Mr. Thomas Marek, a remarkable scientist, whom I respect highly, and learned so much practical skills. Last but not least, many thanks to my research partner Yanxiang Yang, a very talent and good friend, whose contribution is essential to the success and achievement of the team.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Jiang Hu [advisor], Professor Dana Porter from the Department of Biological & Agricultural Engineering, Professor Tie Liu and Professor Peng Li at Department of Electrical & Computer Engineering.

Funding Sources

Graduate study was supported by water seed grant from Texas A&M University.

NOMENCLATURE

AI	Artificial Intelligence
CPU	Central Processing Unit
DDoS	Distributed Denial-of-Service attack
DSSAT	Decision Support System for Agrotechnology Transfer
ET	Evapotranspiration
FAO	The Food and Agriculture Organization of the United Nations
FC	Field Capacity
IoT	Internet of Things
GDD	Growing Degree Days
GUI	Graphical User Interface
MAD	Management Allowable Depletion
MCU	Microcontroller Unit
MDP	Markov Decision Process
NN	Neural Networks
NDFD	National Digital Forecast Database
GPIO	General-purpose input/output
GPR	Gaussian Process Regression
PID	Proportional-Integral-Derivative
PWP	Permanent Wilting Point
ROM	Read Only Memory
RL	Reinforcement Learning
RMSE	Root Mean Squared Error

RAM	Random Access Memory
SVM	Support Vector Machine
TAW	Total Available soil Water (between FC and PWP) in the root zone
TD	Temporal Difference
USDA	United States Department of Agriculture
w.r.t	with respect to

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation

An intelligent or a smart system is characterized by two key attributes: situation awareness and capability of react [17]. Through the use of advanced sensing and communication technologies, today, awareness can be integrated into a system efficiently. Moreover, knowing precisely when and how to react is also essential. Therefore, sophisticated data processing and control techniques are also necessary for any smart system.

Smart technologies have been applied to many fields. One important application is the smart power grid. A power grid is a network that consists of power generators, transmission lines, transformers, and end users. In the past, power grids were one-way systems, from power plants to consumers only. It is difficult to respond to today's ever changing and rising demand. Smart grid utilizes latest sensing, communication, and control technologies, and is more responsive and efficient.

Smart irrigation is another interesting field in smart system research. Like the power industry, agriculture is critically important to our everyday life. Yet, it is also facing tremendous challenge to keep up its production with the fast growing global population and the demand for more and better food. How to better utilize the already stretching water resources becomes the key to solve the food crisis.

Crop yield models like DSSAT (The Decision Support System for Agrotechnology Transfer) [18] and AcquaCrop [7] are very important for agriculture research. Simulations are much faster than field experiments. However, these models don't always guarantee to work for different scenarios. In fact, researchers always use observed data to calibrate their models before carrying on larger scale simulations. It would be better if models can be accurate without the time-consuming calibration process. In addition, models are often used individually, and very few studies have been conducted on model fusion to best utilize useful information carried by different models.

1.2 Problem Formulation

Smart meters installed in communities collect user consumption data and send them back to electricity companies. The data can help planners to prepare electricity production accordingly. An accurate prediction makes the production and transmission more efficient. In this way, companies can ensure their profitability and the users can enjoy low cost electricity. Yet, the integration of renewable energy into smart grid has created some difficulties.

In the past two decades, the European Union (EU) has made a lot of progress in green energy policy making and legislation. The objective is to transform EU's energy supply portfolio towards a more sustainable and environmentally friendly one. Since 2004, the share of renewable energy in gross final consumption in the European Union has doubled, reached 17% in 2016 [19]. In Belgium, 5,468 GWh of electricity, 6.7% of the country's total demand, was produced by wind farms [20]. Encouraged by government's green energy subsidies, local residents are now more willing to buy electricity from renewable sources like wind farms. Residents receive daily guideline price and renewable energy availability. Because customers are sensitive to price change and incentives, guideline price and renewable energy availability may greatly affect actual usage. Yet, the correlations have not been well studied yet.

We obtained a record of 120 days electricity generation from a Belgium wind farm. In addition, a small local community's electricity consumption over the same time period is also collected for our research. Our objective was to provide accurate energy consumption prediction on household and community levels. At household level, the objective was to obtain the peak to average ratio and bill increase for the next day. At community level, the aim was the produce total energy consumption for every hour of the next day.

An accurate prediction is important not only for production planning, but also for security purposes. With the development of smart grid, Internet-based smart devices are deeply embedded. These devices expose power grids to imminent threats from malicious attacks. Some studies have revealed the scale and capacity these types of attacks can have on large power facilities [21]. On the other hand, massive distributed denial-of-service (DDoS) attacks lauched from hacked IoT devices

have already occurred in 2014 [22]. In 2003, a large scale electric blackout paralyzed Midwest and Northeast of United States, even Ontario Canada. It is estimated that over 50 million people were affected, and the total loss was about 4 - 10 billion dollars [23]. For obvious economic reason, most IoT devices have low level or even no protection. There are even YouTube videos showing how to break-down an electric meter and manipulate its readings. Studies have also shown with even limited resource or access to an electric meter, an attacker can still manage to construct a false data injection attack [24]. If energy consumption predictions can be highly accurate and reliable, they can be used as references to identify anomalous activities. Thus, an attack can be detected and stopped at early a stage.

Let's move on to the irrigation. Currently, there are three types of smart controller or control algorithm [25]:

1. Soil moisture sensor based,
2. Evapotranspiration(ET) based,
3. Rain sensor based.

All of these control algorithms have been around for two or more decades. However, none of them have been widely used in commercial market. ET based controllers have received mixed reviews and must be maintained. It requires access to local weather data from an onsite weather station or other appropriate source. Although in some cases, ET is estimated from historical data, such method is not accurate enough for in-season irrigation management. On the other hand, traditional soil moisture controllers are not generally well applied to yield significant water saving. Their designs are based on a bypass circuit attached to a timer. If a soil moisture is higher than a user-adjustable threshold, then the scheduled irrigation is canceled. Recently, a more advanced soil moisture based control is reported [25], known as on-demand control. There is no timer inside the controller, the decision on whether to irrigation is entirely decided by a set of minimum and maximum thresholds. When the soil moisture level drops below the lower threshold, irrigation starts. It stops only when until the soil moisture level reaches higher threshold. Rain sensor based control is the simplest among all three. Rain sensors act as a switch to cut off irrigation when

sufficient amount of precipitation occurs.

All the three methods have some sort of situation awareness and ability to react. Still, we cannot call them highly intelligent or efficient, and each has limitations. Let's consider a situation, where the soil is dry but a heavy rainfall event is expected to come in the following days. A good strategy is to suspend current irrigation plan and wait for the rain to replenish soil water. However, for ET based algorithm, a raining day is just a day with a smaller ET value. It would not react much differently, even if the soil has already been filled up with water after the rain. For soil moisture based and rain based algorithms, only an irrigation scheduled after the rain can be avoided. In many cases, it is already too late. Therefore, we decided to develop predictive capabilities into a control algorithm that reacts to not only real-time soil moisture readings and ET, but also incoming weather events.

In addition, what is the best soil moisture level for a crop at certain growth stage? This question has never been clearly answered. Because of that uncertainty, all the existing smart irrigation technologies use either a threshold or a blurry range to operate. How to set the threshold and the lower and upper thresholds are subject to the users intuition or experience. The concept of Management Allowable Depletion (MAD) has provided a scientific base for the setting of a good threshold. Nevertheless, it is a constant setting across the entire crop season, and too coarse to be optimal.

Finally, crop yield models play important roles in agricultural research. Later, introduced in section 5, our reinforcement learning based algorithm relies heavily on DSSAT simulations. The effectiveness of our design is directly related to the crop yield model's accuracy. Yet, they are not perfect. Sometimes they perform poorly. Different models are developed upon different theoretical bases. They can provide valuable insights at different angles. Therefore, it is a reasonable guess that models may complement each other. Following this logic, the next question is: how? Our answer is model fusion.

Note that, the model fusion techniques we developed can be applied not only in agriculture, but all other research areas that involve modeling and simulation.

1.3 Related Previous Work

For energy prediction, many works have focused on the production side. Solar energy as a common source of renewable energy is relatively easy to predict since its generation is relatively constant. Estimated weighted moving average (EWMA) is often used, and proved to be an effective model under normal weather conditions [26]. For wind energy, boosting-tree and weighted nearest neighbor are also reported [27]. On the consumption side, regression models are used and studied in building energy predictions. However, they are not accurate enough for hourly prediction. Time-series analysis techniques, such as autoregressive integrated moving average (ARIMA) model [28] and autoregressive moving average with exogenous input model (ARMAX) model appear to be good options since human behaviors often have strong temporal correlation. For example, a household that uses a lot of electricity at 7 p.m. on one day might also consume similar amount next day. In another work, a Fourier series model claims to have better performance over time-series analysis. Yet, the assumption is that energy usage is periodic, which might not be true in some cases.

Neural Networks (NN) technique has become a popular prediction method and is used in many applications. There have been some studies of using NN in building energy prediction [29]. Yet, some of these studies work only for a specific building environment. An adaptive neural networks method is proposed by Yang [30], which uses a sliding window to include most recent data and discard old ones. The benefit is that input feature space of the neural networks is manageable, and the temporal correlation can be fully explored. Nevertheless, this method is introduced before the surge of renewable energy, neither does it take guideline price into consideration.

In smart irrigation, wireless sensors can help monitor soil moisture levels in real-time and therefore can provide closed-loop feedback to irrigation control. They have caught a lot of attention in irrigation applications. However, many existing works are restricted to the construction of wireless sensor network and demonstration of its benefit. There is some effort on developing advanced irrigation algorithms making use of wireless sensors and/or weather information. A model predictive control approach is proposed in [31]. During a crop growing season, the control takes current

sensor and weather data as inputs and makes irrigation decisions according to predicted outcome. Although the use of sensor and weather data can largely overcome the deficiencies of traditional irrigation approaches, the prediction still relies on accurate models, which are not always available. A neuro-dynamic programming method is described in [32]. It is essentially a Markov decision process with model based reinforcement learning. Its drawback is the adoption of a linear model, which is an over-simplification of reality.

For topic of crop yield model fusion, there is an unpublished work [15], in which data from 15 global soil, precipitation, elevation, fertilization, irrigation, and harvest database are used to simulated crop yields on six crops around the world, using DSSAT [18] and AquaCrop [7]. Three statistical combination of DSSAT and AquaCrop predictions are built: "Average", "Fitted", and "Corrected". The "Average" is simply taking the mean of two estimates on a country-wide basis. The "Fitted" method used a linear model. Lastly, the "Corrected" method was a quadratic function of the corresponding prediction of the two models' outcomes. Although its performance was good, the work did not provide any specifics on how this quadratic function was constructed.

1.4 Dissertation Structure

This dissertation contains 7 sections. Section 1 Introduction covers general background information, motivation of the researches, and problems we try to solve. It also reviews previous related work. Section 2 provides necessary preliminarily knowledge in machine learning, embedded systems, and agriculture. Section 3 presents a Neural Network-based prediction method of smart community energy consumption. Section 4 introduces a reinforcement learning based irrigation control algorithm. Section 5 reviews technical details on how a real smart irrigation system is constructed. Section 6 focuses on model fusion techniques for crop yield models. Lastly, Section 7 serves as an summary of this dissertation.

2. BACKGROUND

2.1 Introduction of Machine Learning

Machine learning is a very broad field of study in computer science. Machine learning techniques are now widely used in many different fields. Computers can progressively improve their abilities to accomplish tasks without being explicitly programmed. Therefore, with no professional knowledge or rigorous modeling, simply by presenting observed data, computers can gradually "fight" their way through the "unknown" by "self-learning". By skipping complicated and often time-consuming modeling and programming, and delegate the job to machines, productivity in many fields can be dramatically improved.

In general, there are three broad categories that any kind of machine learning techniques can fall in, supervised learning, unsupervised learning, and reinforcement learning.

For supervised learning techniques, data are labeled. We know the inputs and their corresponding outputs. The job is either to find a regression model that fit the data for continuous outputs, or build a classifier for discrete outputs.

On the other hand, the unsupervised learning techniques usually do not enjoy the convenience of knowing everything at the beginning. Therefore, an important task of unsupervised learning is to group data based on relationships among variables.

Reinforcement learning is somewhat unique. It does not have all the knowledge to start with, just like unsupervised learning. However, it has a retro-feed mechanism that enables progressive improvement on the fly. It explores the "unknown" by interacting with its surrounding environment, and has the freedom to choose what action to take in an exploration. The data obtained are evaluated and formed into experience. An algorithm then uses its experience to behave in an direction that maximize long-term rewards.

2.2 Linear Regression

Linear regression is a typical supervised learning technique. We assume the outputs can be fits in a linear model. Depending on the number of variables, a linear regression can be categorized as "univariate" or "multivariate".

A math expression for the model is called hypothesis function. An univariate hypothesis function is:

$$\widehat{y} = h_\theta(x) = \theta_0 + \theta_1 x \tag{2.1}$$

where x is the input, y is the output. $\widehat{y}$ is the estimation of y. θ_0 and θ_1 are parameters that we need to find.

To quantify the quality of our guess, the cost function $J(\theta_0, \theta_1)$ is the following:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\widehat{y_i} - y_i)^2 \tag{2.2}$$

where m is the number of samples, and i indicates the index of a sample data within a training set.

The objective of the training is to minimize the cost function $J(\theta_0, \theta_1)$, ideally to 0.

repeat
  $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$;
  $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$;
until *convergence*;

Algorithm 1: Gradient Descent in Univariate Linear Regression

The most widely used method is the Gradient Descent. The idea is by taking the derivative of the cost function with respect to (w.r.t) the parameters θ_0 and θ_1, we can find the direction of which to reduce error. At each step, we take a small step towards that direction. The speed of learning is controlled by learning rate α. This iterative process is stopped when the gradient is small enough

or the performance is no longer improving. The gradient descent for linear regression can be described by the Algorithm 1.

Multivariate linear regression is an extension of the previous univariate one with a similar mindset. The major difference lays in the use of vectors and matrix.

In multivariate linear regression, the input $x^{(i)}$,indexed by i, has multiple features $x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)}$.

Now, the hypothesis function looks like the following:

$$h_\theta(x^{(i)}) = \begin{bmatrix} \theta_0^{(i)} & \theta_1^{(i)} & ... & \theta_n^{(i)} \end{bmatrix} \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ . \\ . \\ . \\ x_n^{(i)} \end{bmatrix} = \theta^T x^{(i)} \tag{2.3}$$

If we formulate inputs in a matrix form:

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & ... & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & ... & x_n^{(2)} \\ . & . & & . \\ x_0^{(m)} & x_1^{(m)} & ... & x_n^{(m)} \end{bmatrix} \tag{2.4}$$

then, the hypothesis is given by:

$$h_\theta(X) = X\theta \tag{2.5}$$

And the cost function:

$$J(\theta) = \frac{1}{2m}(X\theta - \overrightarrow{y})^T(X\theta - \overrightarrow{y}) \tag{2.6}$$

where $\overrightarrow{y}$ is the vector that contains all y values.

The algorithm for multiple variables is very similar except that the process is repeated for all the features in input data.

repeat

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$

$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$

...

$\theta_n := \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)}$

until *convergence*;

Algorithm 2: Gradient Descent in Multivariate Linear Regression

2.3 Logistic Regression

Although the name "Logistic Regression" may mislead people to believe that it is a regression method, it is actually a classifier.

Let's start with binary classification problems for simplicity. Unlike the linear regression, whose outputs are continuous, the outputs for binary logistic regression are 0s and 1s. As a convention, we call them "labels". 1 as a positive judgment of a sample belongs to the class, and 0 as a negative.

Apparently, since the outputs are bounded value between 0 and 1, a linear hypothesis whose outputs are not bounded is simply not going to work. Therefore, we need a new hypothesis model that cab satisfy: $0 \leq h_\theta(x) \leq 1$. An ideal choice is the "sigmoid function" also known as the "logistic function", taking the form of $g(z) = \frac{1}{1+e^{-z}}$. Nevertheless, we can still maintain the linear combination of all features by making $z = \theta^T x$. The resulting shape is presented on figure 2.1.

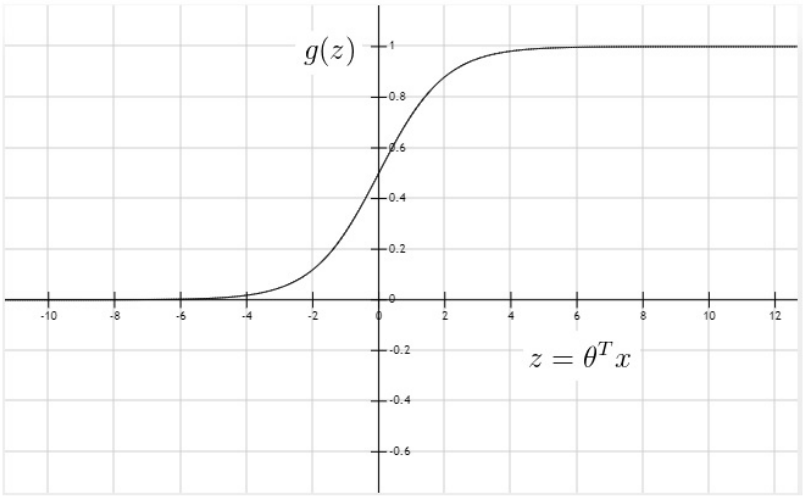


Figure 2.1: Sigmoid function as the form of hypothesis in logistic regression.

Here, in a classification problem, we can intuitively view the hypothesis function as an estimation of the probability that a sample belongs to a class. From the above figure, one can see that whenever $z \geq 0$, the resulting $g(z)$ is greater than 0.5, thus the label is 1.

The cost function $J(\theta)$ in logistic regression is also changed, as shown on figure 2.2:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} C(h_\theta x^{(i)}, y^{(i)}) \tag{2.7}$$

if $y = 1$

$$C(h_\theta x^{(i)}, y^{(i)}) = -log(h_\theta x^{(i)}) \tag{2.8}$$

if $y = 0$

$$C(h_\theta x^{(i)}, y^{(i)}) = -log(1 - h_\theta x^{(i)}) \tag{2.9}$$

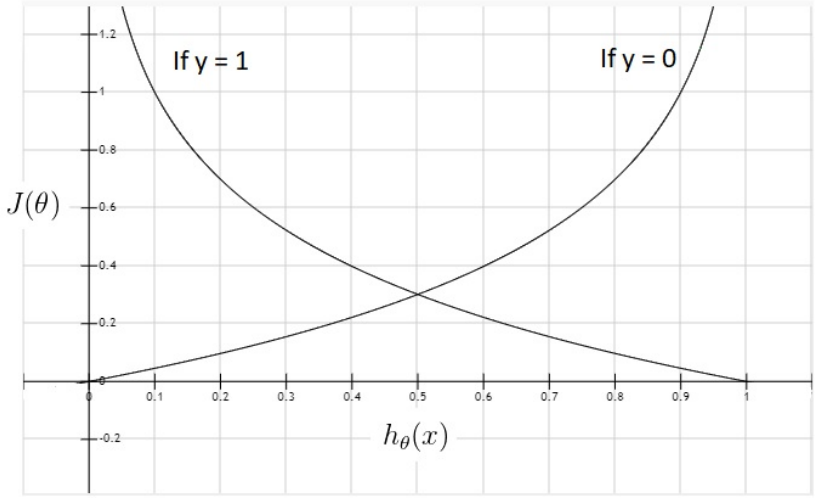


Figure 2.2: Cost function in logistic regression.

The reasons behind such design are:

- Cost function $J(\theta)$ is a strict convex function, which guarantees a global minimum.
- The difference between our "guesses" and "targets" can be reflected in cost function.

As you may find easily, if $y = 1$, the cost increases when the hypothesis deviates from 1. While on the opposite, if $y = 0$, the cost increases as the hypothesis deviates from 0.

It turns out that we can simply combine the two scenarios into one regardless of the value of y.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(h_\theta(x^{(i)}))] \tag{2.10}$$

The last step is the gradient descent. Detailed calculation of the partial derivative of $J(\theta)$ with respect to θ_j is omitted here, with the result presented as following:

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]x_j^{(i)} \tag{2.11}$$

The algorithm for gradient descent used in logistic regression is, in the format, almost identical to the one used in linear regression. However, please note that $h_\theta(x^{(i)})$ is now a sigmoid function rather than a linear.

repeat

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$
$$\ldots$$
$$\theta_n := \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)}$$

until *convergence*;

Algorithm 3: Gradient Descent in Multivariate Logistic Regression

For multiple classification, where the output takes more than two discrete values, the basic logic is the same. We can simply divide the workload into multiple binary classification problems and then concatenate them together.

2.4 Reduce Over-fitting and Use of Regularization

In the real world, datasets are rarely clean. They often come with noise, therefore, contain many outliers. One of these problems is over-fitting, demonstrated by figure 2.3. As shown in Figure 2.3, simpler models, although convenient, lack the ability to achieve high accuracy. One the other hand, by including higher order polynomial terms, more complex models can better separate different classes. Nevertheless, if too much, the existence of several outliers may severely jeopardize the generality. To achieve a balance between accuracy and generality, we introduce the regularization terms into our cost function.

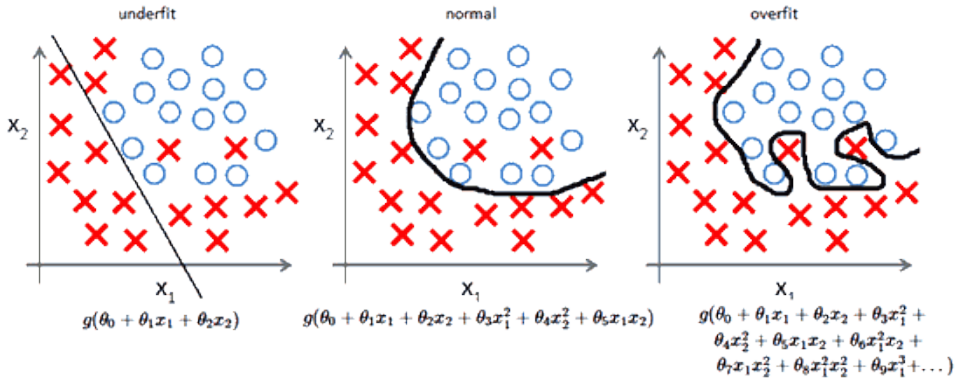


Figure 2.3: Over-fitting when using high order polynomial terms [1].

For example, if we want to reduce the influence of 3rd and 4th order polynomial terms $\theta_3 x^3$ and $\theta_4 x^4$, we can modify our cost function as the following:

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2 \tag{2.12}$$

To eliminate the inflation introduced by the last two terms $1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$, we will need to reduce the θ_4 and θ_3 to very small values close to 0. We therefore, achieve the goal of reducing the influence carried by the 3rd and 4th order polynomial terms $\theta_3 x^3$ and $\theta_4 x^4$, while maintain their existence to tackle complicated shapes.

Similarly, if we want to achieve overall balance between accuracy and generality, we can add the regularization terms for all parameters used in model as a summation:

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \tag{2.13}$$

where, λ is the regularization parameter that adjusts the balance of accuracy and generality.

2.5 Support Vector Machine

Support vector machine is another very powerful supervised learning classification technique that has been widely used. It inherited many traits from logistic regression. Yet, the ideal of maximizing the margin of decision boundary makes such technique more robust.

Suppose we have a 2-dimensional binary classification problem. This green line is called the "decision boundary", which separates two sets of data that belong to two different classes. Let's say the line is described as $\theta^T x = 0$, with a set of parameters $\theta = [\theta_0, \theta_1, \theta_2]$, and $x = [x_0, x_1, x_2]$, where $x_0 = 0$.

Now we can easily make an decision on what class a sample data $x = [x_0, x_1, x_2]$ belongs to by simply comparing $\theta^T x$ to 0. For points in class "o", $\theta_0 + \theta_1 x_1 + \theta_2 x_2 < 0$. For points in class "x", $\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$, as shown on figure 2.4.

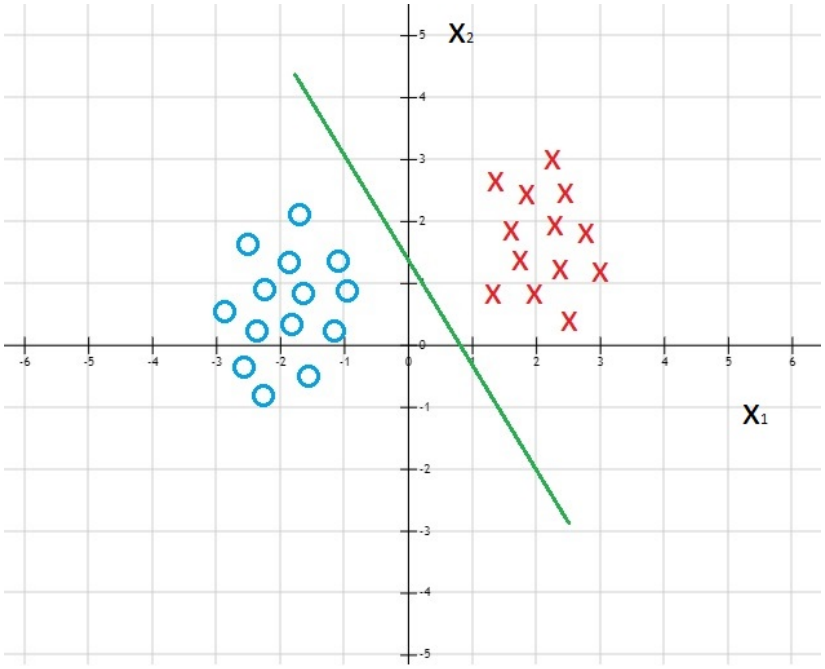


Figure 2.4: Decision boundary for a binary classification problem.

Now, let's switch our view to vector space. The dot product rule of two vectors is defined by:

$$\mathbf{X} \cdot \mathbf{Y} = |\mathbf{X}||\mathbf{Y}|cos\alpha \tag{2.14}$$

where $\mathbf{X} = (x_0, x_1, x_2)$ and $\mathbf{Y} = (y_0, y_1, y_2)$ are 3 by 1 vectors, and α is the angle between $\mathbf{X}$ and $\mathbf{Y}$. The final value of the dot product can also be calculated by:

$$\mathbf{X} \cdot \mathbf{Y} = x_0 y_0 + x_1 y_1 + x_2 y_2 \tag{2.15}$$

If we view the set of parameter θ as a vector Θ, and the decision boundary as another vector $\mathbf{x}$, then since

$$\theta^T x = 0 \tag{2.16}$$

Then,

$$\Theta \cdot \mathbf{x} = 0 \tag{2.17}$$

Note, that the lengths of both vectors are positive, so parameter vector must be perpendicular to the decision boundary. We will come back to this important conclusion later.

Back to our discussion on SVM. Here is some modifications we need to do on the cost function. Firstly, recall that the cost function of a logistic regression is given by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1-y^{(i)})log(h_\theta(x^{(i)}))] \tag{2.18}$$

Now, instead of using the logarithm curve, we replace it with a two-segment strict line, as shown on figure 2.5 and figure 2.6.

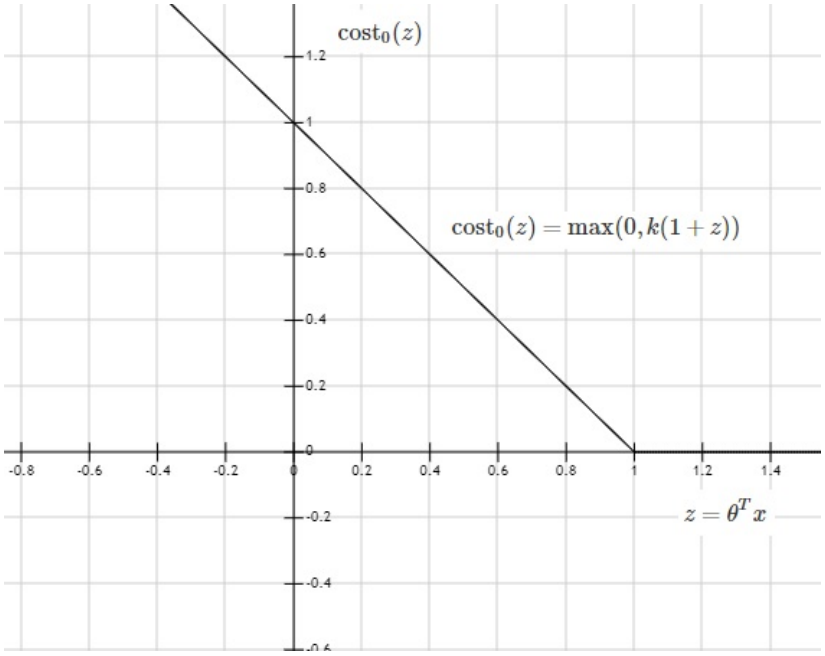


Figure 2.5: Two segment strict line function replaces the original logarithm curve in the cost function when $y = 0$.

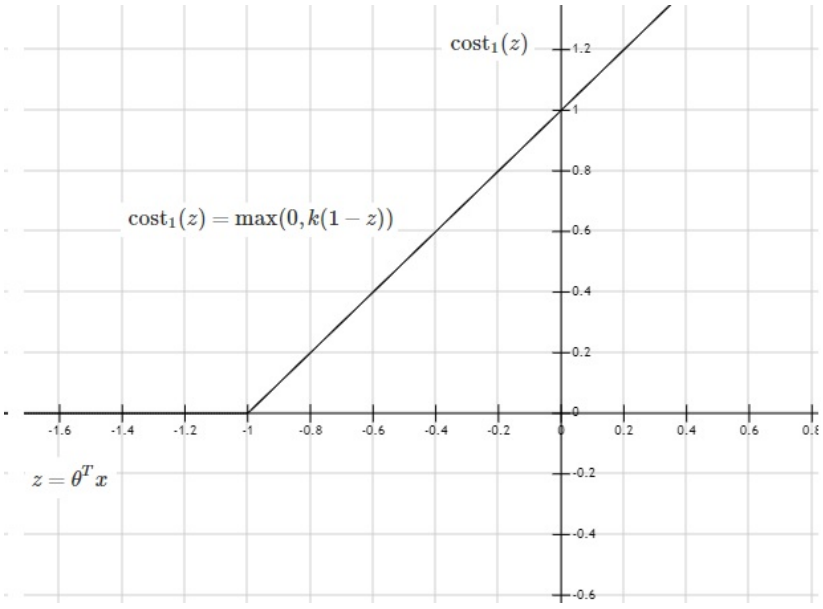


Figure 2.6: Two segment strict line function replaces the original logarithm curve in the cost function when $y = 1$.

The resulting new cost function with regularization is the following:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} cost_1(\theta^T(x^{(i)})) + (1 - y^{(i)}) cost_0(\theta^T(x^{(i)})) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{2.19}$$

By convention, we change the form a little bit to use a C instead of λ.

$$J(\theta) = C\sum_{i=1}^{m} y^{(i)} cost_1(\theta^T(x^{(i)})) + (1 - y^{(i)}) cost_0(\theta^T(x^{(i)})) + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \tag{2.20}$$

The hypothesis of support vector machine is either 0 or 1.

$$h_\theta(x) = \begin{cases} 1, \ \theta^T x \geq 0 \\ 0, \ otherwise \end{cases} \tag{2.21}$$

Finally, we want SVM to give us a boundary that has the maximum margin to both sides. To be more specific, instead of comparing $\Theta^T x$ to 0, we set $\Theta^T x \geq 1$, if y = 1, and $\Theta^T x \leq -1$, if y =

0.

When C is large, the optimization process like gradient descent would eventually make $\sum_{i=1}^{m} y^{(i)} cost_1(\theta^T(x^{(i)})) + (1 - y^{(i)}) cost_0(\theta^T(x^{(i)})) = 0$, leaving the cost function to be:

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \tag{2.22}$$

Recall that the parameter vector Θ is perpendicular to the decision boundary. Therefore, for any point $x^{(i)}$ in the training set, calculating $\Theta^T x$ is equivalent to $p^{(i)} \cdot |\Theta|$, where $p^{(i)}$ is of the projection of $x^{(i)}$ to Θ. Meanwhile, notice that $p^{(i)}$ is also the absolute distance from x_i to the decision boundary. Therefore, in order to make $\theta^T x^{(i)} = 0$, which is equivalent to $|p^{(i)} \cdot |\Theta|| \geq 1$, to be satisfied, we have to make the $p^{(i)}$ as large as possible. Hence, the margin is stretched.

2.6 Neural Networks

The machine learning methods introduced in previous sections may work well for small number of input features and relatively simple relationships. However, many complex shapes require higher order polynomial terms, and when the number of input features is large, it becomes very expensive, in terms of computation, to consider all possible combinations. To be more specific, the number of total polynomial terms is $\frac{(n+r-1)!}{r!(n-1)!}$, where n is the number of input features, r is the number of elements in a combination. For example, input $x = [x_1, x_2, x_3, ..., x_6]$ has 6 features. If we only consider two-element combinations like x_1x_2, x_3x_6, the total number of two-element polynomial terms is 21. If we need to include all the quadratic and cubic terms, the total number of terms will become impractical to implement.

Inspired by the structure of brain neurons, which interconnect with each other, Neural Networks technique was developed. The structure of a typical neural network is shown on figure 2.7.

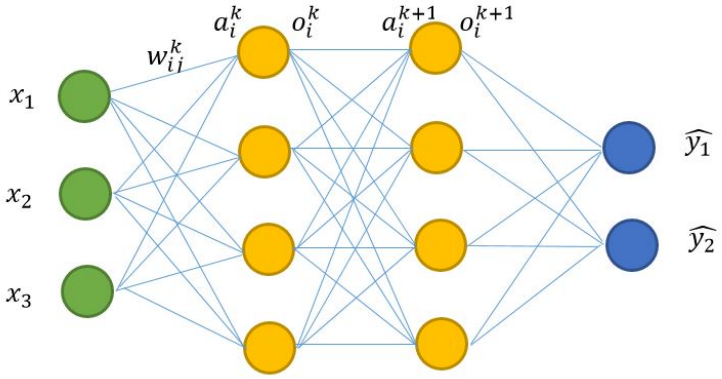


Figure 2.7: Typical structure of a neural networks.

Each circle in the network is called a neuron. Each one is a computational unit that takes in inputs from other neurons in previous layer, processes them, and outputs the result to next layer. The following is a list of definitions that will be used in later math explanation.

- $w_{ij}^{(k)}$: weight connects neuron i in layer $k-1$ and neuron j in layer k.
- $a_i^{(k)}$: product sum for node i in layer k.
- $o_i^{(k)}$: output for node i in layer k.
- $r^{(k)}$: number of nodes in layer k.
- $E(X, \theta)$: error of input X, where θ is the matrix containing all weights in the network.

For instance, the product sum in second layer $a_i^{(2)}$ is calculated by the following equation:

$$a_i^{(2)} = \sum_{j=1}^{r^{(1)}} w_{ij}^{(2)} x_j \tag{2.23}$$

Meanwhile, the product sum in third layer $a_i^{(3)}$ is calculated similarly, except that the weights are multiplied by outputs $o_i^{(2)} = \sigma(a_i^{(2)})$ of the second layer, instead of x_j.

$$a_i^3 = \sum_{j=1}^{r^{(2)}} w_{ij}^{(3)} o_j^{(2)} \tag{2.24}$$

The use of non-linear sigmoid function $\sigma(a_i^{(k)})$ gives the network the ability to model high dimensional relationships. Compared to implicitly listing all combinations of high order polynomial terms, the inter-connecting structure is simpler, but much more powerful than linear regression and logistic regression.

The way to solve the parameters or "weights" of a neural networks is through Backpropagation, an algorithm that uses, again, gradient descent.

Let's first start with some random guess θ. The error function is in the form of mean squared error:

$$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{2.25}$$

The partial derivative the error w.r.t weight $w_{ij}^{(k)}$ is $\frac{\partial E}{\partial w_{ij}^{(k)}}$, and by chain rule:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \frac{\partial E}{\partial a_j^{(k)}} \frac{\partial a_j^{(k)}}{\partial w_{ij}^{(k)}} \tag{2.26}$$

Define the first term on the right as error $\delta_j^{(k)} = \frac{\partial E}{\partial a_j^{(k)}}$, and expand the second term on the right, we have:

$$\frac{\partial a_j^{(k)}}{\partial w_{ij}^{(k)}} = \frac{\partial}{\partial w_{ij}^{(k)}} (\sum_{l=1}^{r^{(2)}} w_{il}^{(k)} o_j^{(k-1)}) = o_i^{(k-1)} \tag{2.27}$$

Thus, partial derivative of $E(X, \theta)$ w.r.t a weight $w_{ij}^{(k)}$ is:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \delta_j^{(k)} o_i^{(k-1)} \tag{2.28}$$

Let's calculate backwards from the output layer. Suppose there is only one neuron in the output layer, and there are m layers in total, and the activation function of the output layer is $g_o(x)$.

$$\delta_1^{(m)} = (g_o(a_1^{(m)}) - y) g_o'(a_1^{(m)}) = (\hat{y} - y) g_o'(a_1^{(m)}) \tag{2.29}$$

$$\frac{\partial E}{\partial w_{i1}^{(m)}} = \delta_1^{(m)} o_i^{(m-1)} = (\hat{y} - y) g_o'(a_1^{(m)}) o_i^{(m-1)} \tag{2.30}$$

To update weights at the output layer:

$$\Delta w_{i1}^{(m)} = -\alpha \frac{\partial E}{\partial w_{i1}^{(m)}} \tag{2.31}$$

For weights that are in the hidden layer immediate next to the output layer, we have:

$$\delta_j^{(k)} = \frac{\partial E}{\partial a_j^{(k)}} = \sum_{l=1}^{r^{(k+1)}} \frac{\partial E}{\partial a_l^{(k+1)}} \frac{\partial a_l^{(k+1)}}{\partial a_j^{(k)}} \tag{2.32}$$

Since error terms in the next layer have already been calculated, we can simply plug them in to form:

$$\delta_j^{(k)} = \sum_{l=1}^{r^{(k+1)}} \delta_l^{(k+1)} \frac{\partial a_l^{(k+1)}}{\partial a_j^{(k)}} = \sum_{l=1}^{r^{(k+1)}} \delta_l^{(k+1)} w_{jl}^{k+1} g'(a_j^k) \tag{2.33}$$

Now, we can put all the pieces together to find the derivative of E w.r.t $w_{ij}^{(k)}$ as:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \delta_j^{(k)} o_i^{(k-1)} = g'(a_j^k) o_i^{(k-1)} \sum_{l=1}^{r^{(k+1)}} \delta_l^{(k+1)} w_{jl}^{k+1} \tag{2.34}$$

Again, we can update weights by:

$$\Delta w_{ij}^{(k)} = -\alpha \frac{\partial E}{\partial w_{ij}^{(k)}} \tag{2.35}$$

Following the same steps, we can roll backward to find all the error terms in each layer, and therefore, be able to compute all the partial derivative w.r.t each weights and update them.

2.7 Markov Decision Process

In 1960, Ronald Howard published the book *Dynamic Programming and Markov Processes*, and systematically explained the Markov Decision Processes. Since then, MDP has been used in many different areas like automatic control, economics, and manufacturing.

The word *Markov* is from Russian, and generally means that given a present state, the future

and the past are independent.

$$P(S_{t+1} = s'|S_t = s_t A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, ..., S_{t-1} = s_{t-1}) = P(S_{t+1} = s'|S_t = s_t A_t = a_t) \tag{2.36}$$

where $P(x|y)$ is the conditional probability of event x given that event y has happened, S_t and A_t are the state and action taken at time t.

Let's formally define some the terms in MDP that will be used in the later context.

- States $s \in S$, which describe various status of an agent,
- Actions $a \in A$, defines what an agent can do to interact with its surroundings,
- Transition $T(s, a, s')$ function returns a probability that an agent will land on state s' from s, by taking the action a,
- A reward function $R(s, a, s')$ calculates the reward for certain state or actions,
- An optimal policy π^*, is a best strategy of what action to take at a certain state,
- Utility $V(s)^*$ is the expected total reward for starting at state s, and thereafter acting optimally,
- $Q(s, a)^*$ is the expected total reward for starting at state s, taking action a, and thereafter acting optimally. Its relationship to other parameters is presented by figure 2.8.

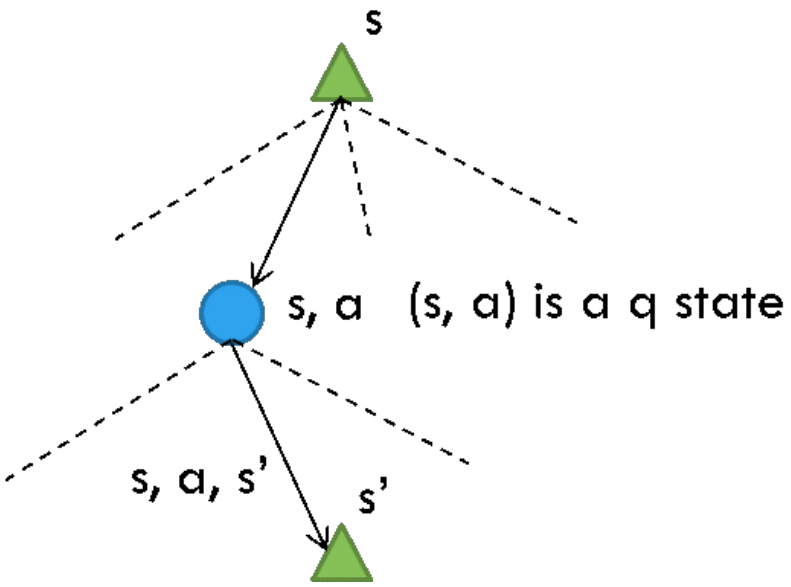


Figure 2.8: The transition from state s to s'. In between the two states is a q state which describes the status of starting from state s and take action a.

MDP provides a math framework to model a stochastic process. In MDP, an agent is the subject. Its status is characterized by states. The current state may change over time. At any time step, an agent makes a decision, takes an action, lands on new state (may be different from the previous state), and receives a reward associated with the state, or the action, or both. Which state the agent lands on is influenced, but not completely, by the immediate action taken by the agent.

Compared to deterministic search problems, in which we search for an optimal sequence of actions, MDP seeks to find an optimal policy $\pi^* : S \rightarrow A$ in a noisy stochastic system. Here comes an interesting question: why the difference? Because, the stochastic nature decides that any fixed sequence of actions when played multiple times, cannot guarantee same result. Different from thinking the process as a whole, the key idea of MDP is to break the process into small pieces. Assuming that if at every time step, the agent takes plays the best "card" at hand, then the final result should be optimal. Here, the short-term benefit is represented by immediate reward function $R(s, a, s')$ and the long-term benefit is described by $V(s)^*$. For different actions, the reward and utility values are summed up, and weighted by the probability of their corresponding transition

$T(s, a, s')$. The agent then checks all possible outcomes and finds the optimal action.

Before we move on to solve the optimal policy, let us first define $V(s)^*$ and $Q(s, a)^*$ mathematically.

$$V^*(s) = \max_a Q^*(s, a) \tag{2.37}$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{2.38}$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{2.39}$$

where γ is a discount factor: $\gamma \leq 1$. The equations are called Bellman equations. There are different ways to solve MDP, one of which is through value iteration (Algorithm 4)

begin
  Initialize $V^*(s)$ with small random values;
  repeat
    for k_{th} iteration, calculate expectimax from each state;
    update $V^*(s)$ by the expectimax for next iteration,
    $V^*_{k+1}(s) := \max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*_k(s')\ ]$
  until *convergence*;
end

Algorithm 4: Value iteration in MDP

An important question is why the convergence is guaranteed? To understand that, we need to first explain why we include discount factor γ in our equations. One easy way of seeing it is that future rewards are worth less than today's value. Let me give you a live example of mine. I have an old great battle buddy *Holt Mica* who owes me several hundreds of dollars of money. One day, *Holt* got paid, and left with sufficient money at hand. The question comes, should I ask him to pay me back immediately or wait. Some of you might be smart enough to say yes. I should ask him to pay immediately. Indeed! If I were to take the money back on that day, I could have used that money to make more on the following days. For instance, deposit in the bank to earn interest, or invest it in the stock market. In fact, I never asked my dear *Holt* to pay back, and the history proves

that he never had enough money to do so in later days. The risk was that the money could be spent for other purposes, or simply lost given a poor management. Thus, the risk associated with future would depreciate the value of today's money.

Back to our discussion on the convergence. Note that $V^*_{k+1}(s)$ is just one more iteration than $V^*_k(s)$. Therefore, the difference is at most $\gamma^{k+1}R_{max}$, which after many many iterations, is very very small, as shown on figure 2.9.

However, such convergence may take a long time. Recall that all we need is a policy not a utility value. So, instead of iterating V values, we can iterate the policy (Algorithm 5). In fact, a policy rarely changes after dozens of iterations. It may converge well ahead of utility function.

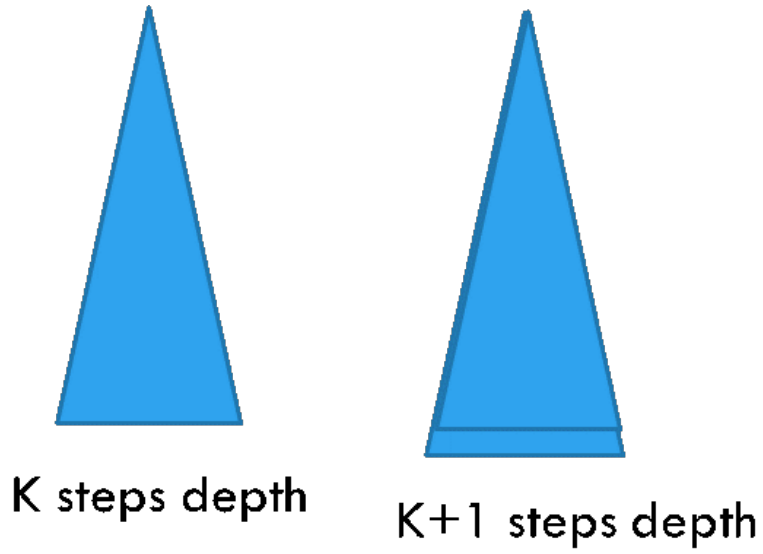


Figure 2.9: K and K+1 steps depth of value iteration

For each iteration, suppose at k_{th},we calculate new utility value with current policy $\pi_k(s)$ for each state s. There are N states and N equations. We can easily solve $V^{\pi}(s)$ as a linear system of equations. Or we can choose to solve it by value iteration, converge but in the order of $O(S^2)$ rather than $O(S^2A)$ in pure value iteration. We use the value of a converged utility function to update the policy. In most cases, policy iteration converges faster than value iteration.

begin
  Initialize π with random mapping;
  repeat
    for k_{th} iteration, calculate expectimax from each state;
    Update the policy using one-step look-ahead
    $V^{\pi}(s) := \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k^*(s')]$
    $\pi(s) := \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k^*(s')]$
  until *convergence*;
end

Algorithm 5: Policy iteration in MDP

2.8 Reinforcement Learning

Reinforcement learning (RL) is somewhat unique in machine learning techniques. Unlike supervised learning, it does not rely on a set of known data. Instead, it is able to "sense" the response from the actions it has taken, and through the "trial and error", it finally reaches an optimal policy. It is not unsupervised learning either, since the reward function of which we model the problem is known already.

The settings of a RL problem are mostly the same with MDP, except that the transition function $T(s,a,s')$ or the reward function $R(s,a,s')$ is unknown. The agent must interact with the environment in order to understand the "rules of the jungles".

There are different ways to solve a RL problem:

- Model Based Learning
- Model-free Learning

The basic idea of a model based learning approach is to learn an approximate model based on experiences obtained from samples, and then, solve for values as if the learned model were correct. To be more specific, although we don't have the true $T(s,a,s')$ or $R(s,a,s')$, we can always estimate from our samples and use the estimation as an approximate model. The rest is the same with standard MDP problem. We can use value or policy iteration algorithms to obtain our optimal policy.

There are two different types of approaches to model-free learning, the passive RL and active RL.

In passive RL, the agent observes outcomes from a fixed policy $\pi(s)$. We can then perform sample-based policy iteration. Previously, in MDP, we update our policy using Bellman equations. However, here in RL problems, $T(s, \pi(s), s')$ and $R(s, a, s')$ are unknown. Yet, we can view $T(s, \pi(s), s')$ as some sort of averaging. The idea is, even without T, as we take enough samples:

$$\begin{cases} sample_1 = R(s, \pi(s), s'_1) + \gamma V^{\pi}(s'_1) \\ sample_2 = R(s, \pi(s), s'_2) + \gamma V^{\pi}(s'_2) \\ ... \\ sample_n = R(s, \pi(s), s'_n) + \gamma V^{\pi}(s'_n) \end{cases} \tag{2.40}$$

The average $V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n} \sum_{i=1}^{n} sample_i$ will be close to the true value of $V^{\pi}(s)$. Nevertheless, we have no prior knowledge about $V^{\pi}_{k}(s'_n)$, in addition, under any policy π, the agent may not be able to visit the same state s again. Thus, we are probably not going to have sufficient samples to make $V^{\pi}_{k+1}(s)$ infinitely close to true value of $V^{\pi}(s)$.

To avoid above difficulty, we can choose to learn from every experience. We call that method the temporal difference (TD) learning. The idea is to update $V(s)$ every time we experience a transition (s, a, s', r), and conduct a running average, as described by the following equations:

$$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s') \tag{2.41}$$

$$V^{\pi}(s) = (1 - \alpha) V^{\pi}(s) + (\alpha) sample \tag{2.42}$$

As the moving average continues, the early samples become less important, and more recent samples gain weight in the calculation.

$$\overline{x_n} = \frac{x_n + (1 - \alpha) x_{n-1} (1 - \alpha)^2 x_{n-2} + ...}{1 + (1 - \alpha) + (1 - \alpha)^2 + ...} \tag{2.43}$$

Initially, our estimations may be very bad, but as the time goes by, initial estimations will weight less, and more recent better estimations will eventually dominate the update. The TD value learning can help us to get V values. But, keep in mind that eventually, what we wanted is a policy. Recall that any policy is extracted from Q values. So a better way would be directly learn Q.

$$sample = R(s, \pi(s), s') + \gamma \underset{a'}{max} Q(s', a') \tag{2.44}$$

$$Q(s, a) = (1 - \alpha)Q(s, a) + (\alpha)sample \tag{2.45}$$

In the above first equation, we replace the V values with Q values, and update $Q(s, a)$ using the same technique in TD learning. In this way, we can learn Q values that are not limited to any certain particular policy, and certainly we do not have to update policy anymore, because from the Q values can be directly used to generate optimal policy.

Note that, in order to make Q values converge, α must gradually decay to zero. The sum of α must go to infinite but the sum of its square must be finite.

We give a particular name for above method: Q-learning.

begin
  Initialize Q table with small random values;
  repeat
    Initialize s;
    repeat
      Choose an action a according to current policy derived from latest Q table;
      Take action a, and observe reward $R(s, a, s')$, and next state s' ;
      Calculate $sample = R(s, \pi(s), s') + \gamma \underset{a'}{max} Q(s', a')$;
      Update Q table by
      $Q(s, a) = (1 - \alpha)Q(s, a) + (\alpha)sample$;
      $s \leftarrow s'$;
    until *s reaches terminal state*;
  until *Convergence*;
end

Algorithm 6: Q-learning

2.9 Introduction of Embedded Systems

Embedded systems can be seen in lots of applications, your cellphone, video recorder, washing machine, or even a toaster. Controls of these electrical or mechanical devices depend on their *embedded electronic systems*, which include one or more microprocessors to provide computational power.

The basic structure of a microcontroller unit (MCU), shown on figure 2.10, has five major parts: clock, central processing unit (CPU), program memory (ROM), data memory (RAM), and input output ports (IO). A clock sets the pace for CPU to run each instruction. The CPU follows instructions stored in ROM, locates and processes data in RAM, and interacts with peripherals through IO.

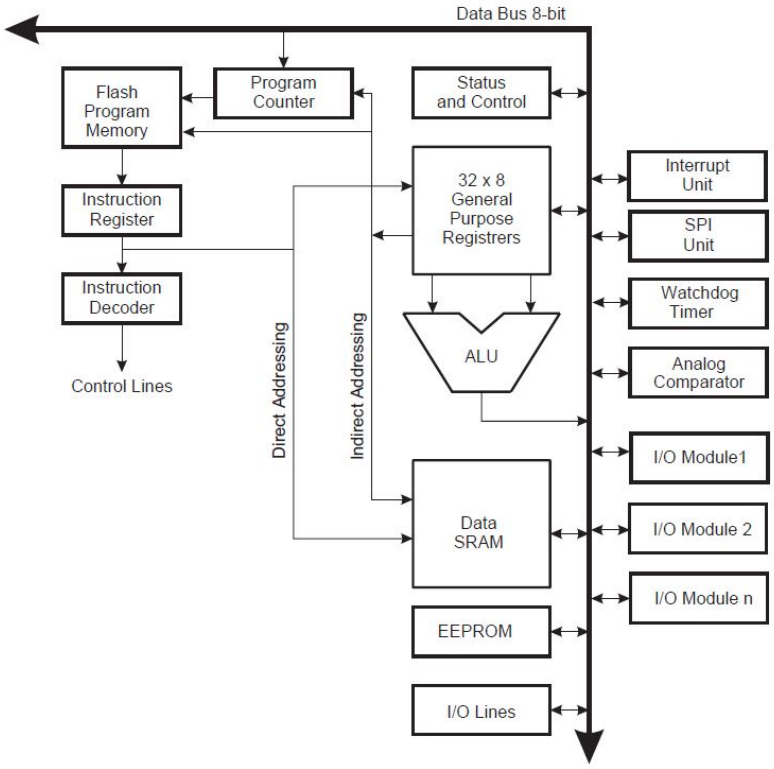


Figure 2.10: Architecture of Atmel 8271 MCU, used by Arduino Uno [2]

In section 5, a detailed introduction is given on how a control system is built for an automated center pivot irrigation machine. Nevertheless, to understand the know-hows, let's first introduce the hardware and software platform on which the control system is constructed.

2.10 Intel Edison

The Intel Edison (figure 2.11) is a small size but powerful computer-on-module that is specially designed for wearable and Internet of Things applications. It contains an Atom 2-Core (Silvermont) @ 500 MHz CPU, 1GB of LPDDR3 memory, 4GB of EMMC storage space. Such strong computational power and the large memory space is very important to support complicated real-time applications. In addition, it has an onboard WIFI module, which can seamlessly connect the MCU to a remote station or any portable smart devices for control and monitoring. Moreover, it enjoys large number of General Purpose Input/Output pins, therefore, can easily interact large number of peripherals. Last but not least, Intel Edison can be programmed in different ways. It supports C, Node.js, and Arduino IDE. It is an ideal hardware platform to build a powerful control system with wireless access capabilities.

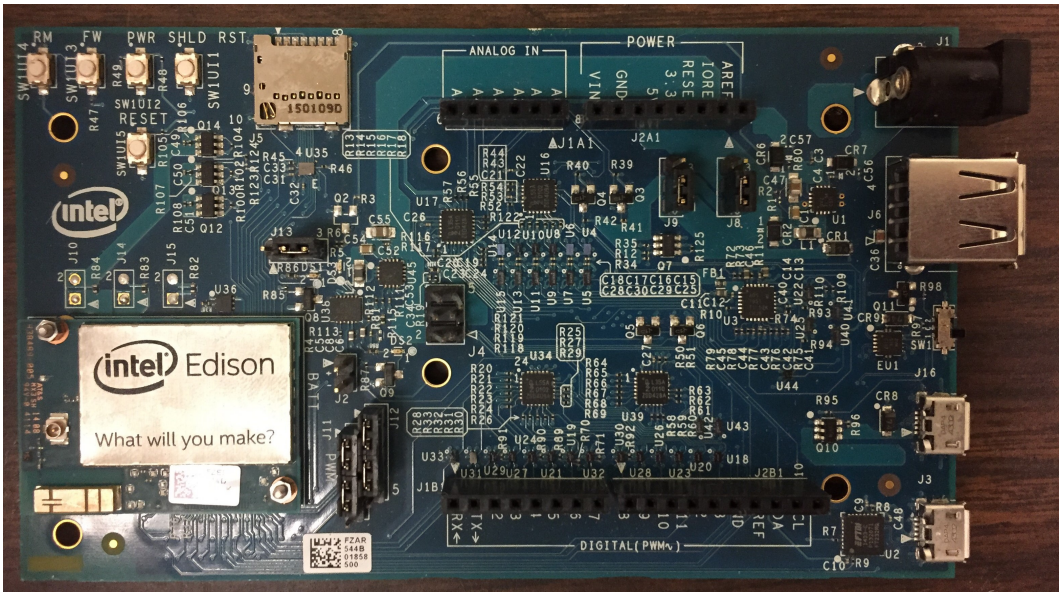


Figure 2.11: Intel Edison board

Table 2.1: Hardware features of Intel Edison [3]

Component	Description
Processor	22 nm Intel SoC that includes a dual-core, dual-threaded Intel Atom CPU at 500 MHz and a 32-bitIntel Quark microcontroller at 100 MHz
RAM	1 GB LPDDR3 POP memory (2 channel 32 bits @ 800 MT/sec)
Internal storage	4 GB eMMC (v4.51 spec)
Power	TI SNB9024 power management IC
Wireless	Dual-band (2.4 and 5 GHz) IEEE 802.11a/b/g/n
Bluetooth*	BT 4.0 + 2.1 EDR
Antenna	Dual-band onboard chip antenna or u.FL for externa
Connector	70-pin Hirose DF40 Series (1.5, 2.0, or 3.0 mm stack height)
Size	35.5 $\times$ 25.0 $\times$ 3.9 mm maximum (to be verified)
Power input	3.15 to 4.5 V
I/O	40 general purpose GPIO
USB 2.0	1 OTG controller
Clocks	19.2 MHz, 32 kHz

An Intel Edison board costs around 50 U.S. dollars, almost twice of the price for its "sibling" Arduino Uno, but it worth every penny. Although a prototype control system is built on Arduino Uno board, the board is simply too "simple" to support more complicated and increasingly demanding tasks that an automated irrigation system requires. It doesn't have enough storage space for large programs to run, and lacks wireless connectivity and Internet accessibility to download, process, store, and interact. Some may argue that after-market wireless and WIFI kits can be found and fit the "holes" (figure 2.12). Yet, the board doesn't have enough extendability to accommodate all jobs at the same time.

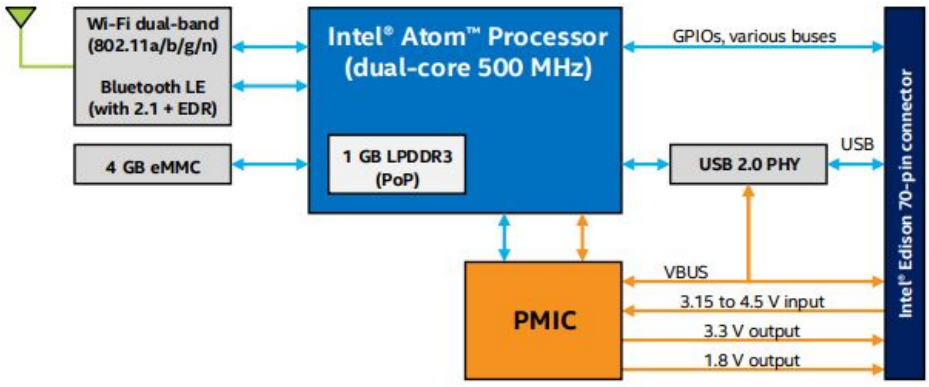


Figure 2.12: Intel Edison board block diagram [3]

In an announcement made on July 5, 2017, Intel officially discontinued its support to Intel Edison platform. The online resources and developers' community will be maintained until June 15, 2020 [33].

2.11 Registers

Registers play important roles in computer architecture. Processor registers are small number of memory cells located inside CPUs, thus very close to Arithmetic Logic Unit (ALU). They facilitate the work of CPU by performing variety of functions like Program Counter (PC), Stack Pointer (SP), Status Register (SR), Constant Generator (CG), and General purpose, etc. Another type of registers, called special function registers (SFR), deserves special attention by embedded system developers, because they control or monitor functions performed by MCUs.

To illustrate how to manipulate a special function register, let's take timerA register in MSP430 MCU as an example, (figure 2.13). It is a 16 bit register. As shown in Figure 2.13, bit 9 to 8 are assigned to control the source of the timer. When configured 01, the low frequency auxiliary clock will source the clock signal to the timer. Bit 7 to 8 are used to control input divider, such way the clock will slow down by a factor of 1, 2, 4, or 8. Bit 5 to 4 control which mode the timer operates in. Bit 2 clears the timer for a new count. Bit 1 can enable or disable interrupt attached to the timer. Lastly, by reading bit 0, we know if an interrupt is pending or not.

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Unused	Bits 15-10	Unused
TASSELx	Bits 9-8	Timer_A clock source select
		00 TACLK
		01 ACLK
		10 SMCLK
		11 INCLK (INCLK is device-specific and is often assigned to the inverted TBCLK) (see the device-specific data sheet)
IDx	Bits 7-6	Input divider. These bits select the divider for the input clock.
		00 /1
		01 /2
		10 /4
		11 /8
MCx	Bits 5-4	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power.
		00 Stop mode: the timer is halted.
		01 Up mode: the timer counts up to TACCR0.
		10 Continuous mode: the timer counts up to 0FFFFh.
		11 Up/down mode: the timer counts up to TACCR0 then down to 0000h.
Unused	Bit 3	Unused
TACLR	Bit 2	Timer_A clear. Setting this bit resets TAR, the clock divider, and the count direction. The TACLR bit is automatically reset and is always read as zero.
TAIE	Bit 1	Timer_A interrupt enable. This bit enables the TAIFG interrupt request.
		0 Interrupt disabled
		1 Interrupt enabled
TAIFG	Bit 0	Timer_A interrupt flag
		0 No interrupt pending
		1 Interrupt pending

Figure 2.13: Timer A register of MSP430 MCU, an example of special function register [4]

2.12 Clocks and Timer

Clocks are very important for computer systems. Normally, MCUs have multiple clocks in order to tailor different demands (figure 2.14). Clock signals may come from an oscillating crystal, or a digital controller oscillator (DCO) circuit. In general, signals from crystals are more accurate than DCO, therefore, often used in high frequency and act as master clock (MCLK). DCO, although relatively inaccurate, is every inexpensive to implement. Thus, so they are often used as auxiliary clock (ACLK) at low frequency. Clock signals are often in the square wave. Generating clock signals is power consuming. Therefore, most modern MCU can switch between full power mode and multiple low power modes depending on different needs.

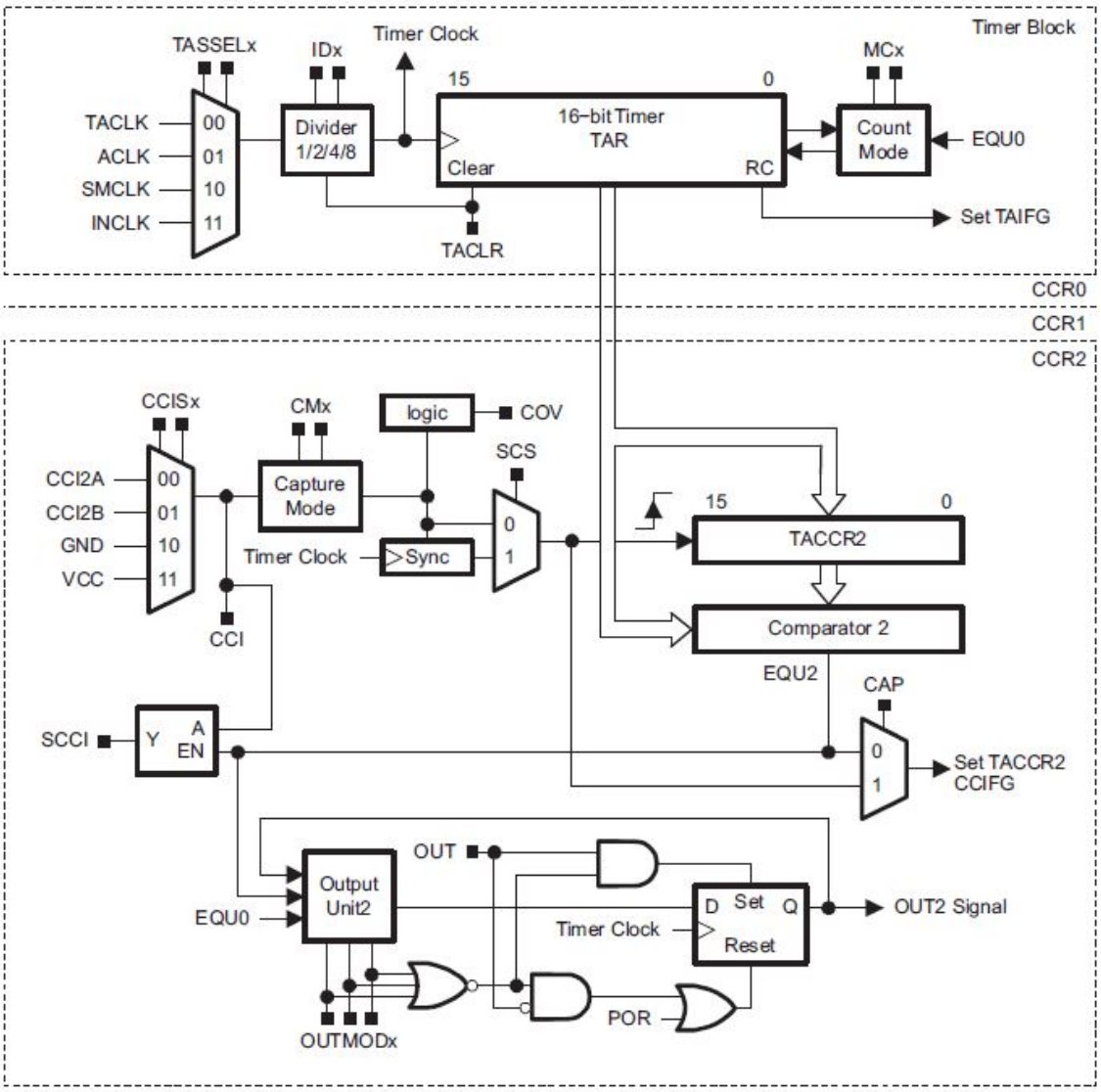


Figure 2.14: Timer A block diagram of MSP430 MCU [4]

Since clock signals are generated periodically, by counting the number of waves and multiply by the clock period, one can estimate time. A timer is basically a counter that counts the clock waves. Figure 2.14 demonstrates a typical structure of a timer. On the top, a multiplexer TASSELx can select different frequency to source the timer. The IDx can adjust the period of a selected clock signal. Then, the TAR is the part of register that stores the count.

Timers in MCU normally have four modes of operation, shown in Table 2.2:

Table 2.2: Typical Timer modes in MCU

MCx	Mode	Description
00	Stop	The timer is halted
01	Up	The timer repeatedly counts from 0 to the value of TACCR0
10	Continuous	The repeatedly counts from 0 to 0FFFh
11	Up/down	The repeatedly counts from 0 up to the value of TACCR0 and back down to 0

2.13 General-Purpose Input/Output (GPIO)

A MCU interacts with its surroundings through IO ports. Intel Edison has 40 GPIO pins, shown on figure 2.15. Most pins are multiplexed with alternate functions for peripheral features, as shown in Figure 2.15. Functions that attached to certain pin can be found on pin diagrams. In Intel Edison and Arduino Uno, pins are grouped to ports. They can be configured individually for output or input direction. When configured as input, voltage can be read. On digital GPIO pins, values are stored as logic 0 or 1 in corresponding bit on data register. On analog pins, values are stored in a register of 1 byte, representing voltage level within a defined range, for example 0-2.5V. When configured as digital output, an output pin can alter its voltage by changing the corresponding bit value on a data register. For Edison and Arduino Uno, the output voltage is 3.3V. If configured as analog output, pins output an analog voltage signal. Pins on Edison and Arduino also have interrupt capability. An interrupt can be triggered by either rising or falling edge, making it easy to connect buttons for controls.

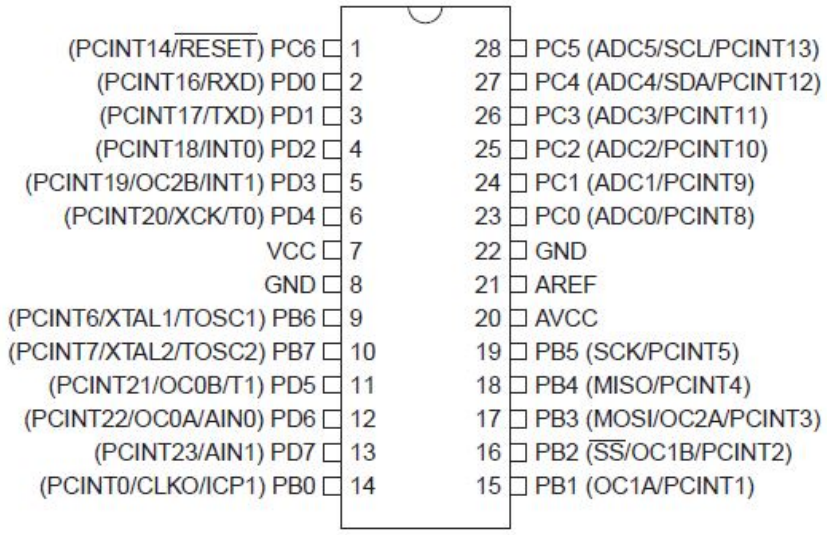


Figure 2.15: Pin diagram of Atmel 8271 MCU [2]

Let's take Atmel 8271, a MCU used in Arduino Uno as an example to illustrate how I/O functions (Figure 2.16). As mentioned before, GPIO pins are grouped into ports. For each port, three I/O memory address locations are assigned to each of three special function registers: Data Register PORTx, Data Direction Register DDRx, and the Port Input register PINx. Writing 1 or 0 on a specific bit of DDRx changes the pin to output or input. When configured as input, writing PINx can enable or disable pull-up resistor attached to the pin. Lastly, if configured as output, changing the value on PORTx can switch the output voltage to either high or low. How the circuits realize above functions can be seen from Figure 2.16.

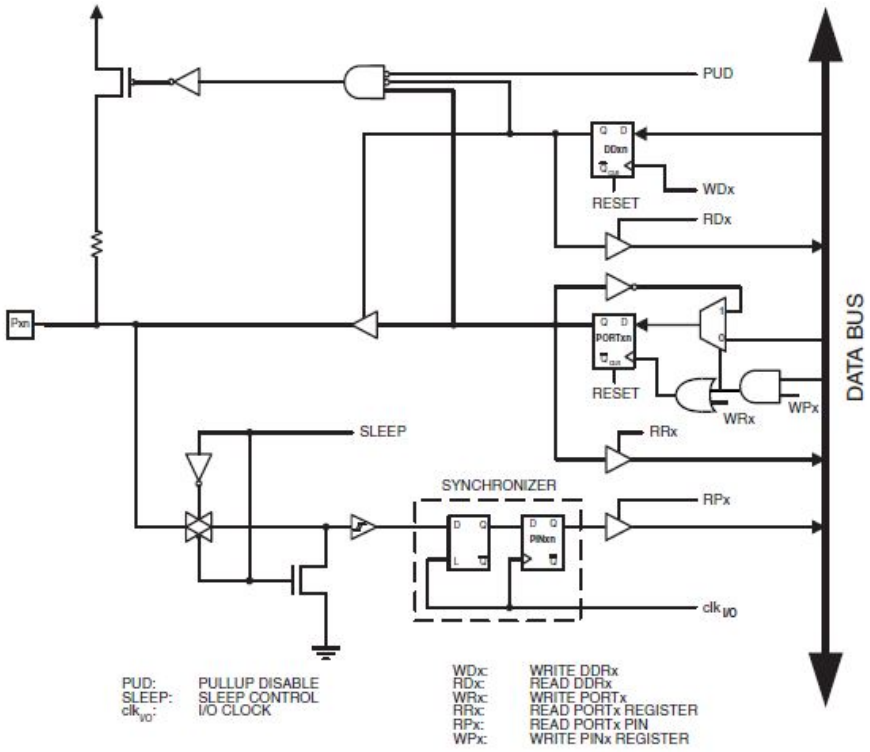


Figure 2.16: I/O Pin Equivalent Schematic in Atmel 8271 MCU [2]

In the design of the smart automated center pivot irrigation system, several GPIO pins are configured to be output pins, connected to Solid State Relays. Their jobs are to control the pivot's speed, direction of moving, and the water pump. To ensure system robustness, beside a web-based control interface, a manual interface is also added. Buttons, switches, and potentiometers provide discrete and continuous control inputs to the system. LED lights, connected to output pins are used to display system status during the operation.

2.14 Polling and Interrupts

In a control environment, a MCU has to respond to events that happen in the system or its environment. There are two ways to handle an event, polling and interrupt.

Polling is a protocol that checks periodically. Developers create a loop that constantly examines the IO status. It is easy to implement. For example, one can compare the input voltage with a pre-defined value, delay for 1 second, and then check again. However, such way, the program

has to routinely halt and perform the checking. Much time is wasted, therefore, it is considered inefficient. Moreover, if a situation only last for very short period of time, it might not be caught. In some cases, missing a critical signal could be catastrophic.

On the other hand, interrupt is a more efficient mechanism to handle unscheduled events. The main program can continue to execute other tasks until an event triggers an interrupt. Then the CPU stops the current task, stores all the necessary data, and responds to the interrupt. A special program called interrupt handler is designed to deal with the event. When the event is handled, the CPU reads back the data and continues the previously unfinished task.

For Intel Edison and Arduino Uno, interrupts can be attached to events like input voltage change, timer overflow, SPI and UART transfer complete, ADC conversion, or analog comparator.

2.15 Mraa, Node.js, and Web Application

Node.js is a JavaScript runtime environment that is used to build web server on Edison board. It is free and incorporated in Intel XDK (figure 2.17) for IoT projects on Intel platforms. Unlike PHP or ASP, which has to do things in order, Node.js is non-blocking event driven. As soon as it sends a task to a computer's file system, it is ready to take on the next job. When the ongoing task is done, the server returns content to the client. In such way, responses in Node.js are significantly faster than in other runtime environment.

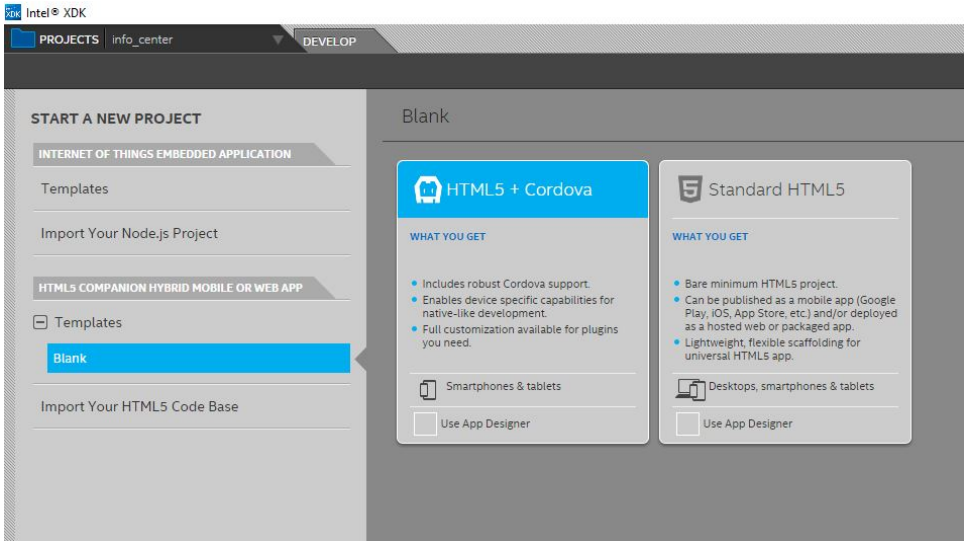


Figure 2.17: Intel XDK is a development environment for IoT and web applications

Fast response capability is critical to the smart irrigation project. In order to meet the requirement of precision irrigation, the center pivot irrigation machine, a giant moving structure, has to accelerate, slow down, or stop at very precise time and locations (within 60 cm). Also, for security reasons, the machine must be able to immediately respond to manual control signals to prevent the machine running into obstacles or damaging structures.

Mraa is a C/C++ library with bindings to C++, Python, Node.js, and Java to interface with the IO on Galileo, Edison, Raspberry Pi, and other platforms [34].

With the help of mraa, manipulating IO are much easier than writing each special function registers. For example, configuring pin 11 on Edison board as the control pin that drives the irrigation machine forward, we can simply write the following code, as shown on figure 2.18:

```
var fwd_pin = new mraa.Gpio(12);   // define gpio pin 12 as output pin of engagement
fwd_pin.dir(mraa.DIR_OUT);         // set the gpio direction to output
var fwd_State = 0;                  // store the control signal as variable fwd_state
fwd_pin.write(fwd_State);          // write the control signal to the output pin
```

Figure 2.18: Configure and write output signal to Edison using mraa library.

2.16 Introduction of Agriculture

Agriculture provides food, materials, and medicines that sustain the survival and prosperity of our human society. Two agriculture related projects are to be discussed in section 5 and 6. This discussion serves as an introduction for people who have no agricultural background.

2.17 Soil and Water

Soil is the medium for plant to grow. It is a mix of minerals, organic matter, gases, liquids, and countless organisms (figure 2.19). Although it seems to be compact, there exist tiny spaces in between, called pores. When dry, those spaces are occupied by air. However, when soil receives water, these pores are filled with water.

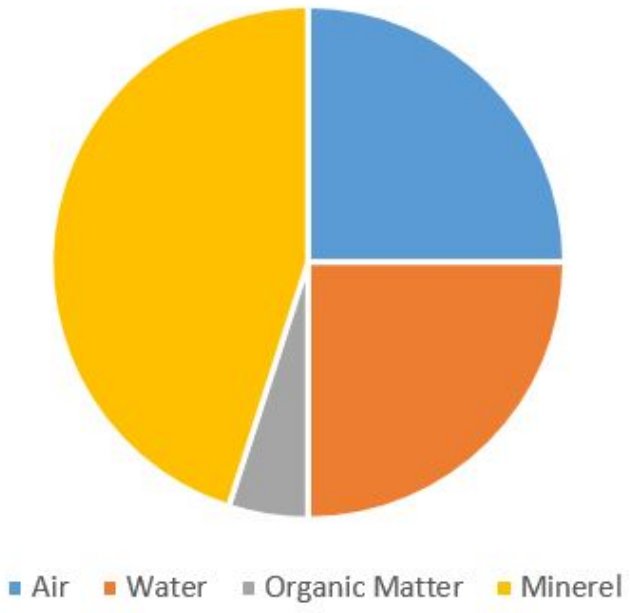


Figure 2.19: Approximate composition of soil by volume.

There are four major types of particles in soil (figure 2.20), they are categorized by sizes:

- Gravel: $\geq$ 2.0 mm in diameter.
- Sand: 2.0 to 0.05 mm.

- Silt: 0.05 to 0.002 mm.
- Clay: $\leq$ 0.002 mm.

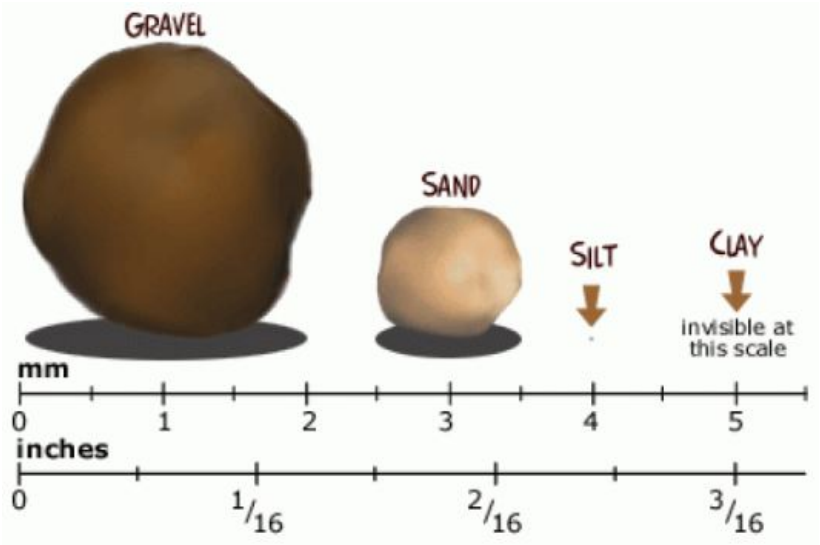


Figure 2.20: Size of gravel, sand, silt, and clay [5].

Many different soil classification systems are developed, and the most widely used one is the USDA textural classification (figure 2.21), adopted in 1938 [6]. Based on the different composition of three major particles, sand, silt and clay, soil is further refined to 9 more classes:

- sandy clay
- silty clay
- silty clay loam
- clay loam
- sandy clay loam
- loam
- sandy loam
- silt loam
- loamy sand

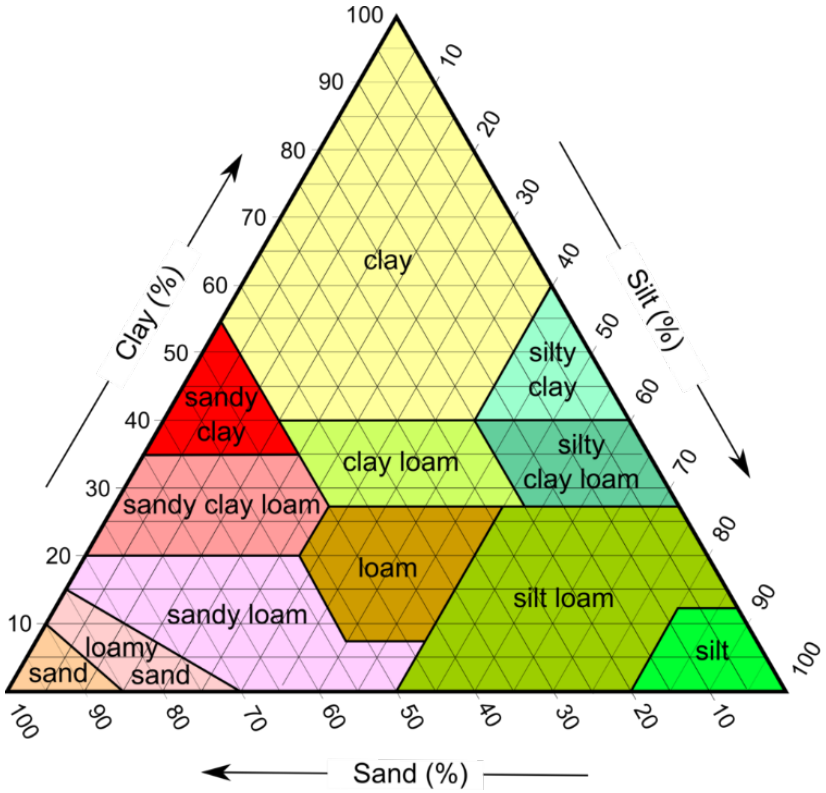


Figure 2.21: USDA soil textural triangle [6]. Soil type is determined by the composition of clay, silt, and sand

Table 2.3: Influence of Soil Texture Separates on Some Properties of Soils [14]

Property/behavior	**Sand**	**Silt**	**Clay**
Water-holding capacity	Low	Medium to high	High
Aeration	Good	Medium	Poor
Drainage rate	High	Slow to medium	Very slow
Soil organic matter level	Low	Medium to high	High to medium
Decomposition of organic matter	Rapid	Medium	Slow
Warm-up in spring	Rapid	Moderate	Slow
Compactability	Low	Medium	High
Susceptibility to wind erosion	Moderate (High if fine sand)	High	Low
Susceptibility to water erosion	Low (unless fine sand)	High	Low if aggregated , otherwise high
Shrink/Swell Potential	Very Low	Low	Moderate to very high
Sealing of ponds, dams, and landfills	Poor	Poor	Good
Suitability for tillage after rain	Good	Medium	Poor
Pollutant leaching potential	High	Medium	Low (unless cracked)
Ability to store plant nutrients	Poor	Medium to High	High
Resistance to pH change	Low	Medium	High

Plants cannot live without water. 80% - 95% of plant's protoplasm is constituted of water. Water is an essential part of photosynthesis and the very media that carries nutrients. In addition, plants need water to supply turgidity in order to keep themselves in proper position.

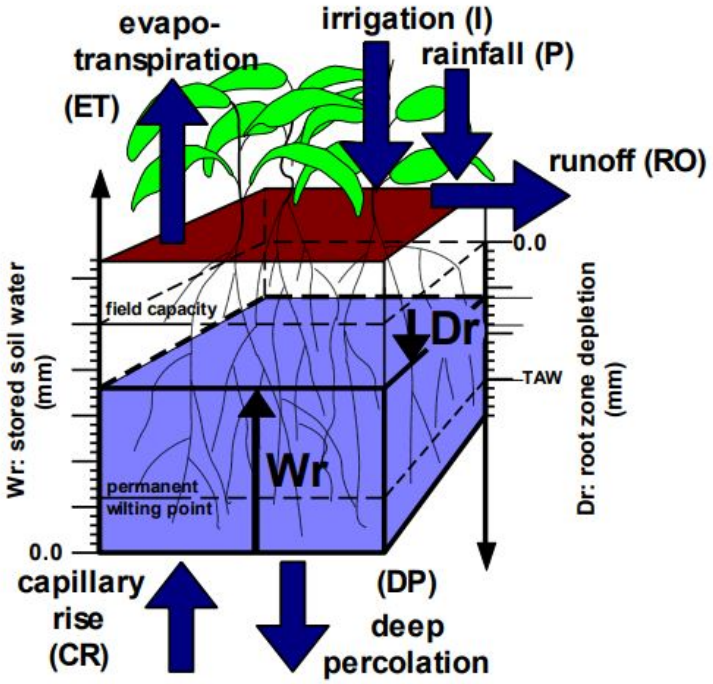


Figure 2.22: Soil water balance [7].

Let's view soil as a reservoir or, simply, a water tank (figure 2.22). The process of water seeps into soil is called infiltration [7]. In agriculture, water is supplied by either rainfall or irrigation. On the other hand, there are four ways that water is lost from the tank: surface evaporation, crop transpiration, runoff, and deep percolation. If the influx is too strong, for example, it rains heavily, then not all the water can be retained. Part of the water could be lost due to surface runoff. Further, due to gravity, some water escapes root zone soil's grasp, and percolates into deeper layer. Such phenomenon is called deep percolation. However, when the upper layer of soil becomes dryer, water is transported from deeper layer upward to replenish the loss. Such process is named capillary

rise.

The rate of infiltration is strongly related to soil texture. If it is coarse, therefore contains large pores, water can infiltrate rather fast. On the contrary, fine textured soil is hard to infiltrate.

The amount of water held in soil is expressed by soil moisture content, in either depth or percent of volume. For example, a soil moisture content of 200 mm/m is equivalent to $\frac{200mm \times 1m^2}{1m^3} \times 100\% = 20\%$ volumetric water content.

There are three levels of water balance in the soil moisture content that deserve particular attentions: saturation, field capacity (FC), and permanent wilting point (PWP).

At saturation, all the pores are filled with water. Soil reaches its maximum water holding capacity. However, the state of saturation cannot stay long. When water is excessive, gravity gains upper hand, deep percolation occurs. After some time, deep percolation gradually decreases and eventually stops. A temporary balance is restored. At this point, soil reaches its field capacity. Then, after some time, water is gradually lost due to surface evaporation and crop transpiration. At certain point, plants can no longer extract enough water for survival. As a result, the plant wilts, and the soil moisture level at such point is defined as permanent wilting point.

Here, we also define total available water (TAW) as water content at FC minus water content at PWP, as shown on figure 2.23 and figure 2.24.

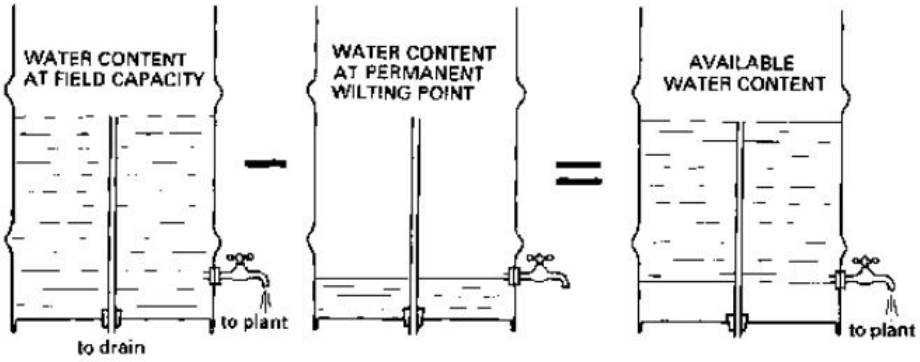


Figure 2.23: Total Available Water (TAW) is the amount of water available in the soil for plant growth [8].

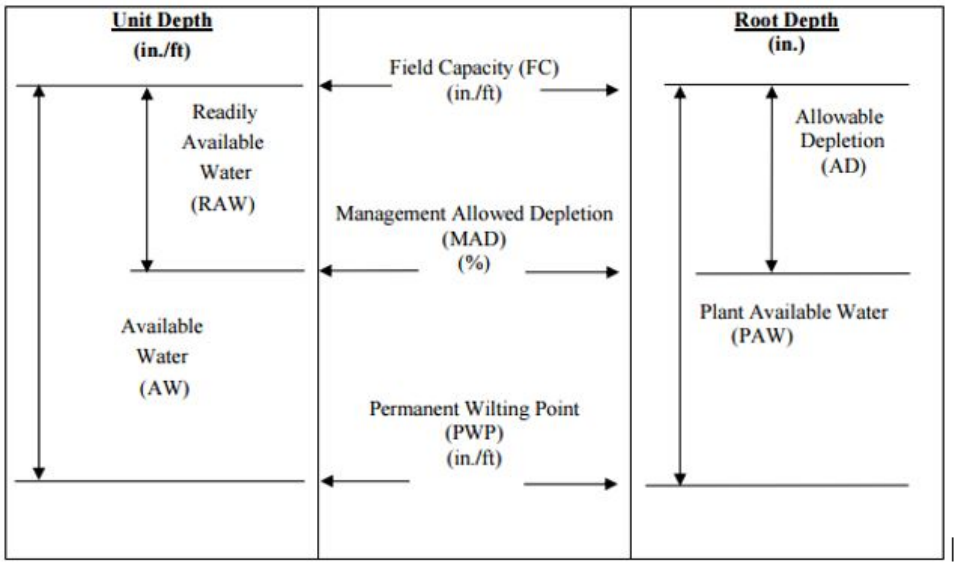


Figure 2.24: Relation of different soil water terminologies [9].

In 1966, John. L. Merriam developed a concept called Management Allowable Depletion (MAD) so that time and depth of water application in irrigation can be evaluated. This concept established a model to relate maximum soil water deficiency to economic value of water, labor, and crop value. In this concept, MAD refers to the maximum amount of Plant Available Water (PAW) allowed to be removed from the soil before irrigation refill occurs [35]. Here, PAW refers to the total amount of water held in the plant root space.

For irrigation, engineers must answer the question of when and how much water shall be applied to the field. The idea of MAD is to irrigate before the soil water is depleted to a level that will cause drought stress on the crop. MAD helps to stimulate deep root growth, and preserve water.

In agricultural research and practice, people often consider the water loss due to evaporation and transportation all together, and designated a special term Evapotranspiration (ET) to it. We often use depth per unit time to express ET rate, which is the loss of water depth in unit time. For example, for a land of 10,000 m^2, an ET rate of 1mm/ day means the total loss of the water on that day is $1mm \times 10,000m^2 = 10m^3$. By doing such calculation, farmers are able to find out how much water shall be replenished in the next round of irrigation, and plan the water use accordingly.

Many factors affect ET, like weather parameters, crop characteristics, management and environmental aspects. Radiation, air temperature, humidity and wind speed are four major factors in ET variance under weather parameters. Evaporation process requires energy to convert liquid water to gas. Strong radiation and high air temperature delivers energy to the surface soil and facilitates the process. However, high humidity level results in small vapor deficit, which suppresses evaporation. Strong wind drives gradually saturated air away and helps to maintain large vapor deficit, thus helps in increasing ET. Crop factors, like different crop height, reflection, ground cover, and roots characteristics affect ET as well. Managerial difference also accounts for ET variance. For example, subsurface drip irrigation system applies water close to plant roots while keeping surface soil dry. Thus, ET is significantly lower than other irrigation methods.

In order to conveniently relate ET demand of the atmosphere to local water factors, "Reference crop evapotranspiration" (ETo) is introduced. It is the ET value of a hypothetical grass land (abundant of water on surface), serving as a reference value.

The reference ETo can be calculated by the following FAO Penman-Monteith equation [13]:

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma\frac{900}{T+273}u_2(e_s - e_a)}{\Delta + \gamma(1 + 0.34)u_2} \tag{2.46}$$

where ET_o is the reference evapotranspiration [$mm\ day^{-1}$], R_n is net radiation at the crop surface [$MJ\ m^{-2}day^{-1}$], G is soil heat flux density [$MJ\ m^{-2}day^{-1}$], T is air temperature at 2 m height [C], u_2 is wind speed at 2 m height [$m\ s^{-1}$], e_s is saturation vapour pressure [kPa], ea is actual vapour pressure [kPa], $e_s - ea$ is saturation vapour pressure deficit [kPa], Δ is slope vapour pressure curve [kPa C^{-1}], γ is psychrometric constant [kPa C^{-1}].

A more detailed review will be given in section 6.

One can calculate ET associated with a type of crop using a "crop coefficient (K_c)" applied to relate reference crop ET_o to crop ET (ET_c).

$$ET_c = K_c \times ET_o \tag{2.47}$$

Note that K_c is specific to: Crop type, Growth stage, ET model and reference crop, and local conditions. ET can also be measured by monolithic weighing lysimeters (figure 2.25 and figure 2.26), which is a giant but very sensitive container on a scale.

Figure 2.25: Surface appearance of a monolithic weighing lysimeter.

Above figure shows what a weighing lysimeter looks like from the surface. Mr. Thomas Marek, second on the left, was introducing the renowned large monolithic weighing lysimeters at Bushland during our first visit in December 2015. Thomas Marek himself is the designer and maker of the system. Dimensions of the soil box are about 2 m x 2 m x 1.5 m depth. People from left to right are: Dr. Gary Marek, Mr.Thomas Marek, Dr. Jiang Hu, Mr. Yanxiang Yang, the author, and Dr. Charles Hillyer.

The underground structure of a Lysimeter is massive. It is filled with soil and embedded with rest of the field. ET is measured by mass balance:

$$\Delta mass_{measured} = Irrigation_{known} + Rain_{measured} - drainage_{measured} - ET \tag{2.48}$$

Figure 2.26: A corner of a lysimeter underground structure.

2.18 Crop

A crop is a plant or animal product that can be grown and harvested extensively for profit or subsistence [36]. Crops are very important to the survival of human species. Important agricultural crops include: rice, wheat, soybean, potatoes, sugarcane, etc.

Many factors may affect the growth of crops, soil water, air temperature, soil fertility, and salinity, etc.

Let's first discuss the effect of soil water. Plants need water to supply turgidity in order to keep themselves in proper position. Water flows from low solute, high water concentration to high solute, low water concentration. In cells of a plant, a lipid bilayer cell membrane permits the flow of water but not the solute. Therefore, water rushes into the membrane and eventually, the pressure pushes the cell's wall to form a status of turgid. We call this pressure turgidity or turgidness. Turgor pressure keeps plants upright and stiff.

When water is insufficient, we say the crop is under drought stress. Drought stress affects canopy development as well as root expansion. If crop is severely stressed, it can result in failure of pollination and canopy senescence. The stress response can be reflected in ET estimates as stress coefficients K_s. It can take different forms, linear, convex, and logistic.

Soil water at less than field capacity may be beneficial (figure 2.27). Sometimes, limited stress at right time and duration, can stimulate deep root growth. Research has shown that less frequent but deep watering helps to stimulate plant root to grow deeper into the soil. Such approach has less evaporation loss since top soil is only wet during first hours or days of irrigation. It also leaves room for more oxygen. On the other hand, frequent light watering makes plants grow shallow roots and leave less room for oxygen. More importantly, a lot more water is wasted due to run off, deep percolation, and evaporation.

K_s represents a relative reduction in ET by crop casued by stress. The reduced ET generally results in loss of nutrient uptake (since nutrients are taken up with water), diminished photosynthesis and therefore diminished carbohydrate formulation. Leaf senescence further reduces the plant's ability to produce carbohydrate. These factors result in lower plant biomass, and lower harvestable

yield.

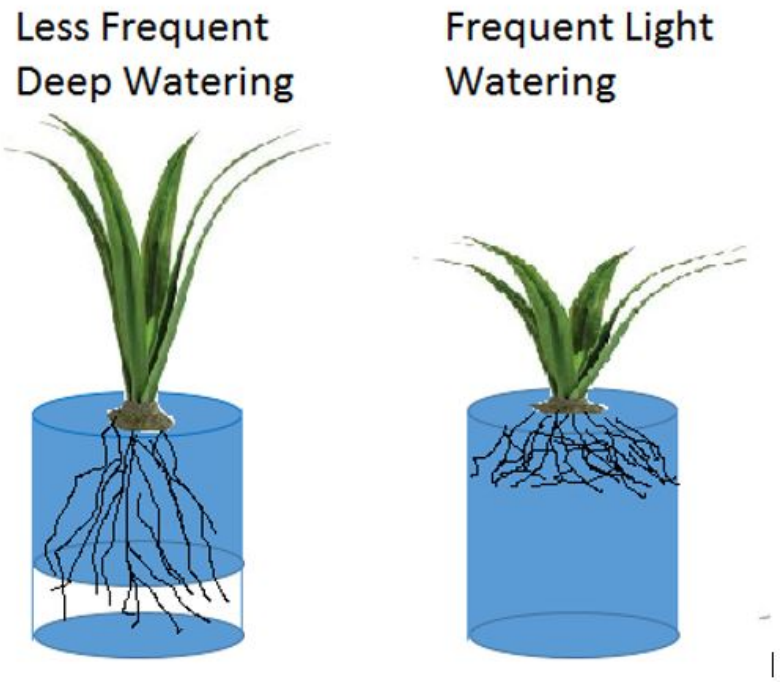


Figure 2.27: Plant growth with different water application.

Air temperature can also largely influence the growth of crops. If it is too cold, the growth might completely stops. We often track crop growth by growing degree days (GDD). It is defined by equation 3.3:

$$GDD = T_{avg} - T_{base} \tag{2.49}$$

where, T_{base} is the base temperature, below which crop development stops. T_{avg} is the average temperature of a day. GDD are accumulated each day that $T_{avg} > T_{base}$.

Soil fertility is another factor to consider. Sufficient nitrogen in the soil can help the canopy development. Phosphorus is very important root development and reproduction processes like pollination, seeding, and fruit development. Potassium is critical to stem growth, transportation, and reproduction processes.

Lastly, if soil salinity, measured by average electrical conductivity of saturation soil-paste extract (ECe) from the root zone, is too high, biomass production is reduced.

2.19 Irrigation Management

Irrigation has been a key practice in agriculture for over 5000 years. It refers to the artificial means to supply water to lands during periods of inadequate rainfall. Irrigation fosters plant growth and may also be used to suppress weeds (in rice production) and to protect plants from frost.

Different types of irrigation methods can be categorized, in general, into 4 different groups.

- Surface irrigation
- Microirrigation
- Irrigation using sprinkler systems
- Sub-irrigation

Surface irrigation, also called flood irrigation, has been the most common irrigation method in the world. It floods the surface of irrigated lands so that water can cover, and infiltrate into the soil. Apparently, this method is less controllable, and large amount of water is wasted if not proper managed. In microirrigation, water is applied through pipelines with low pressure. Methods like surface drip irrigation, subsurface drip irrigation, and micro-sprinkler irrigation belong to this category. They are normally every expensive to implement as pipelines have to be embedded into the field, and lines have to be carefully maintained.

Irrigation using sprinkler systems is widely used in today's industrialized agriculture. Water is distributed from high or low pressure sprinklers, which are mounted on overhead structures. Sprinklers can be designed to spin when water is running through. This makes the application very uniform. Sprinklers can also be installed on moving platforms so that with very limited cost, one system can serve the entire field. Two most widely used configurations are center pivot irrigation system and linear (lateral move) irrigation system. A center pivot system has a center tower and the system moves in circles around the center point. Water is pumped through the pipe from the center point, and released along the radius. A linear system moves straight and applies water to the field as it scans. It is cheaper to build but far more expensive to operate.

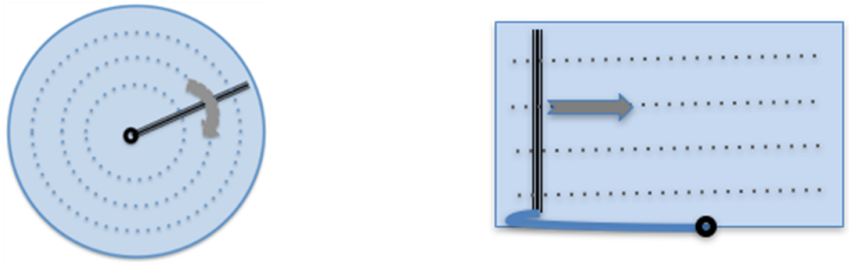


Figure 2.28: Center pivot and linear move sprinkler irrigation systems.

Center pivot systems move in a circular pattern, figure 2.28a. The arrow and dashed lines indicate travel of wheeled towers. Water is supplied to the lateral at the pivot point. Linear move sprinkler systems move in a straight-line pattern, figure 2.28b. The arrow and dashed lines indicate travel of wheeled towers. A flexible hose connects the lateral to the water source.

In section 5, a smart control system that is mounted on a Valley Low Pressure center pivot irrigation machine (figure 2.29) is introduced. A center pivot machine moves like a compass with one end tied to the center pivot point and rest of the structure moves around the pivot point. It has several spans, each is roughly 60 meters long. The structure is supported by towers at the end of each span. The mobility is provided via the wheel powered by motors installed on each tower. As a machine moves, water is pumped to the pipeline and released along the radius. Towers don't move at the same time. The outer most tower moves first, and the next tower down the line detect the angular difference and start to move in order to align with the outer most tower. Same steps take place in all towers down the line in sequence. The material to build the center pivot pipe and trusses is usually galvanized steel or aluminum to withstand tough environment against corrosion. Special epoxy lined or HDPE-poly lined pipes are used for more corrosive waters.

Figure 2.29: The center pivot system in our project.

One interesting question is how to ensure uniformity in center pivot irrigation. Since movement is a circle, the area covered by different sections of the machine various with the radius. That means if the water is released with the same rate everywhere, then inner areas receive more water per unit area than outer areas. To solve this problem, nozzles are designed that sizes are smallest at the inner spans and increase with distance from the pivot point. With proper design and installation, high application uniformity can be achieved.

The center-pivot irrigation system sees wide use in the United States, Australia, New Zealand, Brazil, Sahara region, and the Middle East. Its major advantages over other types of irrigation system are lower operating cost and higher water application efficiency.

Another common one is sub-irrigation, in which water is delivered from below by raising the water table, and being absorbed upwards. It is often seen in greenhouse, river valleys or permanent grassland in lowlands with high water tables.

Irrigation water comes from surface water, groundwater, or other non-conventional sources

like treated waste water. Water scarcity is becoming worse over time. Since 1960s, the world's population has doubled and we are taking more water from rivers and underground than any time ever before, causing problems like environmental degradation and ground subsidence. Irrigation is the single most water consuming activity in the world, but also critical to the survival of human species. We shall do our best to improve irrigation efficiency and save water for our next generations.

2.20 Weather Forecast

Weather is extremely important to agriculture; temperature affects the speed of crop growth, solar radiation provides the energy for photosynthesis, precipitation supplies necessary water. In section 5, we connected our irrigation control system to the Internet, and programmed it to download real-time weather data and near-term forecast from the National Digital Forecast Database (NDFD). An algorithm is developed to deal with different kinds of weather conditions and optimize farmers' profit. Here, let me briefly explain some terms and database that I will use in later sections.

To manage agricultural activities to the best of our abilities, an accurate weather prediction is essential. However, atmosphere is chaotic. Although modern computer technologies have unprecedented computational power, errors still exist, and sometimes very large. Ensemble forecast, illustrated by figure 2.30, compares a collection of predictions and picks the most likely prediction.

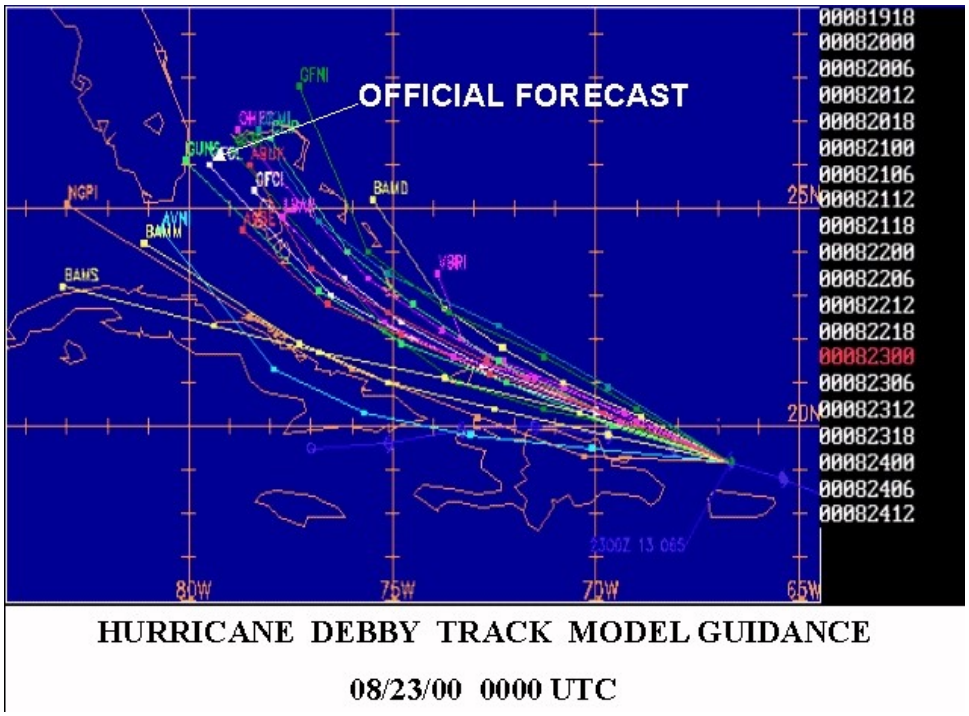


Figure 2.30: Ensemble prediction of Hurricane Debby's path in 2000 [10].

The generation of a deterministic numerical weather forecast involves three steps: acquiring initial state using observation data, stimulating evolution from initial state, and processing the data to calculate results. Note that each forecast is obtained by a single initial state. Yet, the result is never perfect. There are four major reasons [10]:

1. Equations used by a model do not fully capture processes in the atmosphere,
2. Model resolution is not sufficient to capture all features in the atmosphere,
3. Initial observations are not available at every point in the atmosphere,
4. The observational data cannot be measured to an infinite degree of precision.

The Quantitative Precipitation Forecast (QPF), shown on figure 2.31 and figure 2.32, is the expected amount of precipitation a specific area will have over a specific period. QPF is normally provided in two different formats, a quantitative format, which forecasts amounts, and a qualitative format, which forecasts the probability of a specific amount basis.

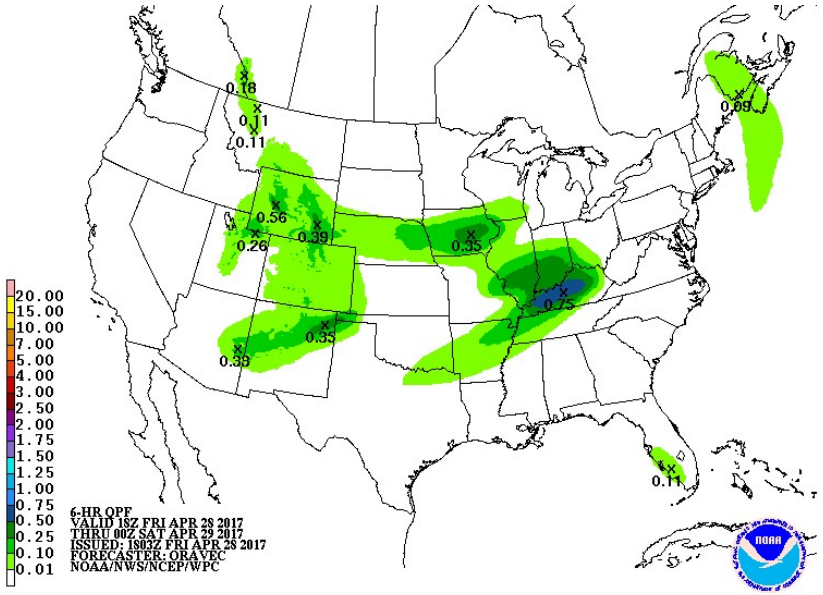


Figure 2.31: 6 Hours Quantitative Precipitation Forecasts from 1800 April 28 2017 to 0000 April 29 2017 [10].

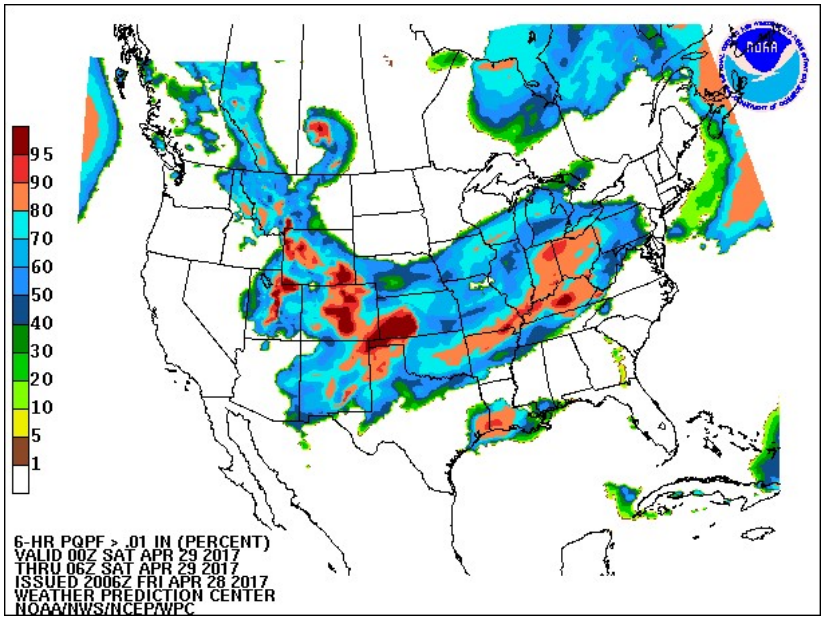


Figure 2.32: 6-Hour Probabilistic Precipitation Prediction of United States from 0000 April 29 2017 to 0600 April 29 2017 [10].

Table 2.4: Data Content of NNDC Climate Database Global Surface Summary of Day

STN	Station number
WBAN	Weather Bureau Air Force Navy number
YEAR	The year
MODA	The month and day
TEMP	Mean temperature for the day in degrees
DEWP	Mean dew point for the day in degrees
SLP	Mean sea level pressure for the day
STP	Mean station pressure for the day
VISIB	Mean visibility for the day
WDSP	Mean wind speed for the day
MXSPD	Maximum sustained wind speed reported
MAX	Maximum temperature reported during the day
MIN	Minimum temperature reported during the day
PRCP	Total precipitation (rain and/or melted snow)

Historical weather data are of great importance in simulation. In section 5, historical weather data are used to create a simulation environment, where computer learns what is the optimal action to take under different weather and water stress conditions. In section 6, historical weather data of around 200 stations around the world are collected and fed into two of the most widely used agricultural simulation software to compare their prediction accuracies. Model fusions are performed using different machine learning techniques that utilize both results to produce better predictions.

The database used in our project is NNDC Climate Database, maintained by National Oceanic and Atmospheric Administration, under U.S. Department of Commerce. The database contains up-to-date weather record for over 10,000 stations all over the world. Although quality and completeness vary by countries and stations, it is a very reliable source of data.

3. COMPARATIVE STUDY ON NEURAL NETWORK-BASED PREDICTION OF SMART COMMUNITY ENERGY CONSUMPTION [1]

3.1 Introduction

Managing the supply of power and predicting future electricity usage have long been a challenge for the power industry. The major difficulty comes from the complicated role each contributing factor plays in the load profile. With the aid of the recent development in advanced metering infrastructure, utility companies are able to collect massive real time data. In the smart home infrastructure, a guideline price is generated by the utility to facilitate the energy usage scheduling of the customers. In particular, the guideline price is high at the peak hours to discourage the energy usage there, which can help balance the energy load in the power grid. Figure 3.1 shows a guideline price and energy load of a small community over 24 hours. These data enable us to build effective algorithms and tools to predict future trend of energy consumption. With high quality prediction, utility companies can make better decisions on designing pricing policies for profit improvement while maintaining the energy consumption balance.

In general, research work in this area can be categorized to two classes: top-down and bottom-up. The top-down approach disregards end users, and focuses on macro economy level. It uses Gross Domestic Product (GDP), unemployment, and inflation as inputs to regress national level energy consumption. The bottom-up approach estimates energy consumption at household level and uses it as an estimation to assess consumption in a larger scale. The engineering and statistical methods are two major techniques used in bottom-up approach [37]. Engineering methods [38] need to consider all possible contributing factors, not only require large inputs, but also suffer from unspecified constituents. Statistical methods [38] such as Conditional Demand Analysis estimate consumption directly without prior assumptions of user specific knowledge. However, the models are not flexible [39], and their performance is limited by multicollinearity.

[1] Part of this section is reprinted with permission from "A Comparative Study on Neural Network-based Prediction of Smart Community Energy Consumption" by L.Sun, J.Hu, Y.Liu, L.Liu, and S.Hu, 2017. In Proceedings of the 2017 IEEE 3rd International Conference on Smart World Congress, pp.311-318, ©2017 IEEE.

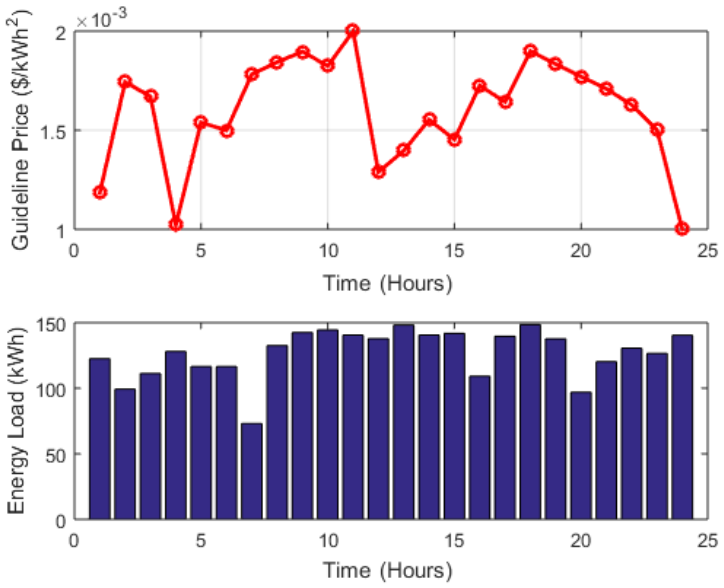


Figure 3.1: Guideline price and energy load in a smart home infrastructure.

In recent years, there are some attempts to use NN methods to model the residential energy consumption. NN enjoys many advantages over traditional methods. They are easy to build and have good flexibility. Previous comparative study shows that decision tree and NN are viable approaches to capture patterns in energy consumption [40][41]. Among those attempts, some use the building features including transparency ratio, insulation thickness and orientation as inputs to predict building heating energy requirements [42]. While the prediction accuracy is high, such an approach requires prior knowledge about certain buildings, and is unable to show dynamic changes over time or any userś behavior patterns. In some other works, NNs are built to predict energy consumption based on temperature data [43][44]. Nevertheless, accuracy of those works is not very high since temperature is not the only contributing factor.

We built and compared different NN architectures to predict energy consumption at household and community level, using data collected from advanced metering devices. The objective of our work is to find the best way to produce high quality prediction and provide insight for improving prediction accuracy for future research. The summary of our contribution is as follows:

- Our approach is highly accurate. Cross-validation results are above 99.4%.

- We address dynamic change of energy consumption over time and have the flexibility to deal with change of environment.
- Performance of DNN, NN with large number of neurons, and NN with small number of neurons are compared. The result has shown that DNN structures are not necessarily better than simple structures.
- The prediction is independent of building structural information, and only based on easily accessible data, making it applicable to apply in large scale at low cost.

The idea is to use NN to model the relationships between contributing factors and future energy consumption. Traditional NNs are used to predict bill increase and peak to average ratio (PAR). In addition, experiments are conducted to see how much improvement can be achieved by applying data pre-processing, adding more neurons, and building deep structures. More importantly, temporal relationship is explored by considering past energy consumption history.

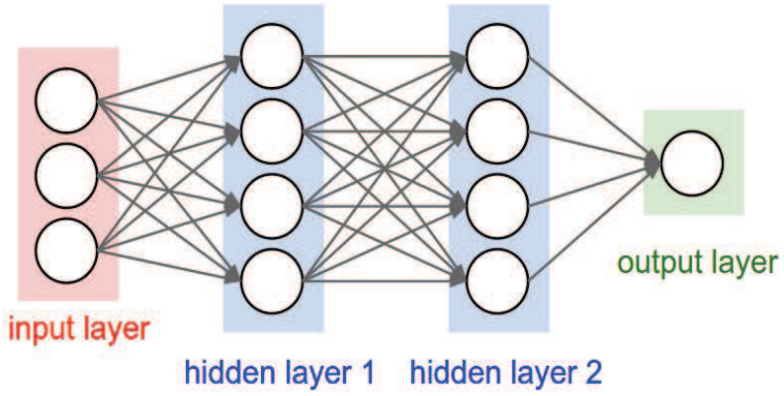


Figure 3.2: Structure of 4 layers NN.

To review, we known that NNs are very good at finding and learning underlying patterns. Their structures and the way they process input data are very similar to the neurons of human brain. Figure 3.2 shows a typical structure of NN. In NN, there is a set of pairs (x, y), and the goal is to find a function that accurately maps x to y [45]. Here, x is called an input, and y is called a target.

Neurons in adjacent layers are connected to each other. Each connection carries a "weight". The output h_i, of neuron i is,

$$h_i = \sigma(\sum_{j=1}^{N} W_{ij}x_j + B_i) \tag{3.1}$$

where $\sigma()$ is an activation function, N is the number of input neurons, W_ij is the weight, x_j is the input value. B_i is a constant offset.

The weighted sum of all connections to a neuron at hidden layer is fed into the activation function. It brings non-linearity to the relationship and the ability to model complicated correlation. There are many choices of activation functions. The most commonly used ones are:

$$Logistics : f(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

$$Tanh : f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{3.3}$$

$$Softsign : f(x) = \frac{x}{1+|x|} \tag{3.4}$$

$$Rectified\ linear\ unit :\ f(n) = \begin{cases} 0 & for\ \ x < 0 \\ x & for\ \ x \geq 0 \end{cases} \tag{3.5}$$

$$Softmax : \sigma(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, for\ j = 1, ..., K \tag{3.6}$$

NNs are to be trained for minimizing the error between the estimated output values and the true output values (target values). A loss function is calculated from output values and the target values to measure errors. The most widely used training algorithm is Backpropagation[46]. It searches "the minimum of the loss function in weight space using the method of gradient descent"[47]. Gradient descent is an iterative algorithm that finds local minimum of a function by stepping toward the opposite direction of the gradient.

3.2 Prediction Method

A list of notations is given as follows:

D: Any given day in a time series

t: Any given hour in a time series

E: Energy Consumption

G: Guideline Price

N: Sliding window size, the number of previous data points

R: Regression value

RE: Renewable Energy Availability

X: input vector, consisting of elements of factors that affect the future energy consumption

X': Normalized input vector, feeding into NN

Y: Output vector, the output vector of NN

Y': Target vector, the known energy consumption

$PAR(d)$: Peak to Average Ratio of day, defined as:

$$PAR = \frac{max(E_{t1}, E_{t2}, ..., E_{t24})}{Average(E_{t1}, E_{t2}, ..., E_{t24})}, \tag{3.7}$$

where $t_1, t_2, ..., t_{24}$ are 24 hours in d.

$BI(d)$: Bill Increase,

$$BI(d) = Bill(d) - Bill(d-1) \tag{3.8}$$

MSE: Mean Square Error

We measure the performance of our prediction by Mean Square Error (MSE) and regression R.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2 \tag{3.9}$$

$$R = \sqrt{1 - \frac{\sum_i (Y_i - Y_i')^2}{\sum_i (Y_i - \frac{1}{n}\sum_{j=1}^{n} Y_j')^2}} \tag{3.10}$$

MSE measures the difference between outputs and targets, the smaller, the better. R ranges from 0 to 1, and measures how a dependent variable is predictable from the independent variable(s). The closer it is to 1, the more accurate our prediction is. Below, R is used to measure accuracy.

At household level, our objective is to model correlations between the guideline price $G(d)$ $G(d-1)$ and a household́s $PAR(d+1)$ and $BI(d+1)$ in the next day. As shown in Figure 3.3, after training, one can use our network to predict $PAR(d+1)$ and $BI(d+1)$ from Guideline Price.

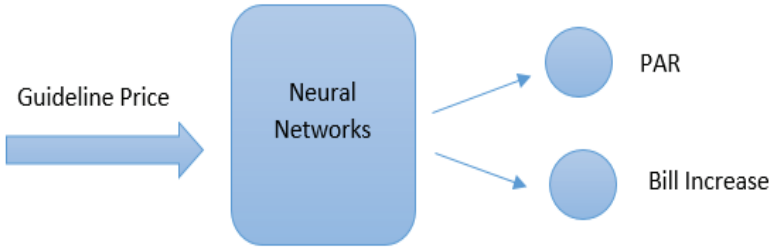


Figure 3.3: Prediction of PAR and Bill Increase using Guideline price.

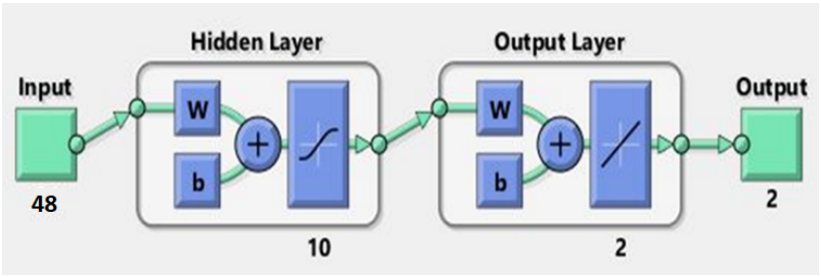


Figure 3.4: Prediction of hourly energy consumption is generated on three pieces of information $G(t)$, $E(t-24)$, and $RE(t)$.

At community level, our goal is to predict energy consumption at time t. Prediction of community level energy consumption requires more information. Besides price, other factors may also affect future energy consumption. One such factor is availability of renewable energy. In

recent decades, renewable energy is playing an increasingly important role in our energy profile. Thus, one adds the availability of renewable energy to our input as well. Apparently, history of energy consumption carries important information for future prediction. For example, electricity consumption of a residential area tends to increase significantly around dinner time as people are off work and return back to their homes. It is obvious that energy consumption at hour t today provides a useful reference on how much energy will be consumed at $t+24$ hour tomorrow. Based on this observation, for predicting energy consumption at hour t, $E(t-24)$ is used as an element of input. As shown in Figure 3.4, there are three contributing factors $G(t)$, $E(t-24)$, and $RE(t)$.

The quality and representation of data are quite important in machine learning because data can be noisy. A recent study has shown that gradient descent converges more efficiently with data pre-processing [46]. Data pre-processing is also called feature scaling in machine learning. Typical data pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc [48]. Normalization is conducted as follows:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}. \tag{3.11}$$

There is no difference in what sequence training data are fed to the network. The underlying assumption is that all data points are independent. However, it is certainly not true for a time series since considerable temporal information is lost. To address the above problem, SWNN is proposed.

A SWNN is a NN that takes previous target values as elements of input. We know recent history carries important information about future energy consumption. How much one wants to consider is controlled by the size of the sliding window N. The prediction of community level energy consumption at a given hour t is set to be a function of contributing factors $X(t)$ stated in Figure 3.4, and recent history: $E(t-1), E(t-2), , E(t-n)$, the prediction function is defined as below.

$$Y(t) = f(E(t-1), E(t-2), , E(t-n), X(t)). \tag{3.12}$$

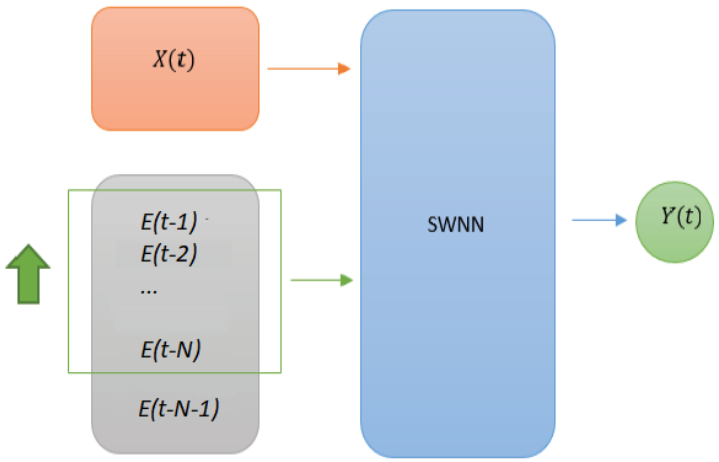


Figure 3.5: A SWNN considers a pre-defined length of the recent history of a time series. The window "slides" forward as the prediction moves on to next point in line.

As shown in Figure 3.5, a window of size N slides forward to include most recent history of Y. Figure 3.6 shows how a SWNN works. The algorithm starts with loading data files and forming input and output pairs. For hour t, N most recent history of energy consumption are added as elements of the input vector. If $t < N + 1$, all available previous historical records of energy consumption are included, and 0s are padded to make the number of elements to be N. Subsequently, the other contributing factors are concatenated to the input vector. The input is then fed to the network for training. When the training for time t is done, one slides the window forward for the next hour $t + 1$, and repeat the process for all t until convergence.

3.3 Evaluation and Comparative Study

The data set used in this experimental setup contains the energy load of a community, guideline price and renewable energy generation for 120 days. The renewable energy generation is obtained from the Belgian wind farm data [49], which are scaled to match the reasonable energy generation of a community level wind turbine. Note that the community aims to minimize the electricity bill for utility energy usage, the energy load data are generated using the smart home scheduling algorithm in [50], given the guideline price and renewable energy prediction. In return, the guideline

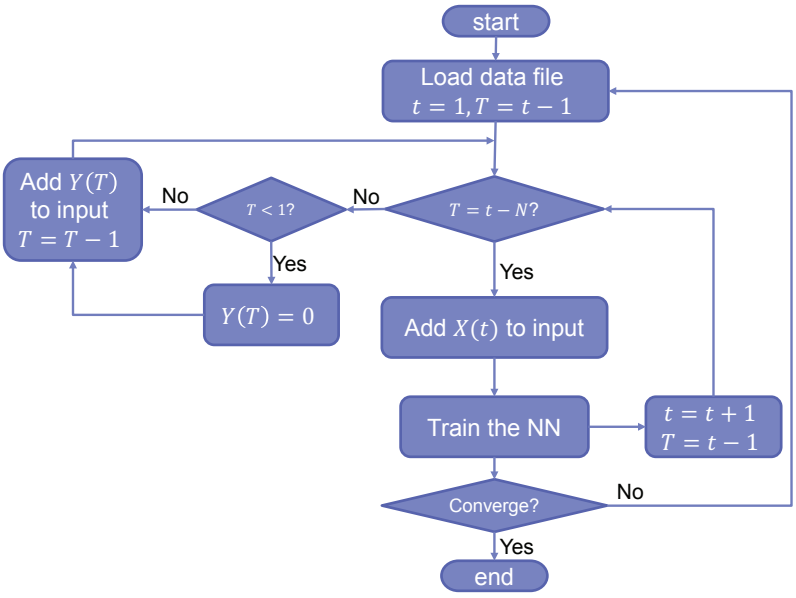


Figure 3.6: SWNN algorithm.

price of next day is generated based on the energy demand and availability of renewable energy today.

Conventional NN and SWNN are constructed by MATLABś machine learning tool box [48]. DNN structures are constructed by Googleś Tensorflow library [51]. We divide 70% of the data for training, 15% for validation, and 15% for testing. During the training phase, the network adjusts its weights and biases on training data, decides when to stop training on validation data, and evaluate the performance on test data.

We randomly choose a household from the community, and pick a period of 101 days. The record of the guideline price in that period is the input of the NN, the PAR (Equation 7) and BI (Equation 8) of the next day are our targets. For each day, input is a 24 by 1 vector, and output is 2 by 1 vector.

Figure 3.7 shows the internal structure of the NN. There is only one hidden layer in our structure, 10 neurons in the first layer and 2 in the output layer. We choose logistic function, defined by Equation 2, as our activation function. The optimization function is the Levenberg-Marquardt backpropagation [52][53].

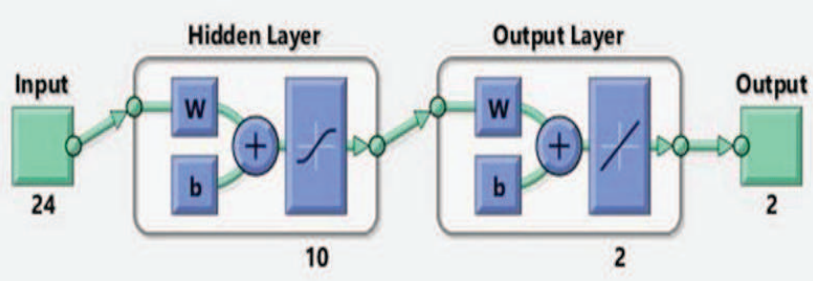


Figure 3.7: Configuration of NN for predicting PAR(d+1) and BI(d+1) at household level.

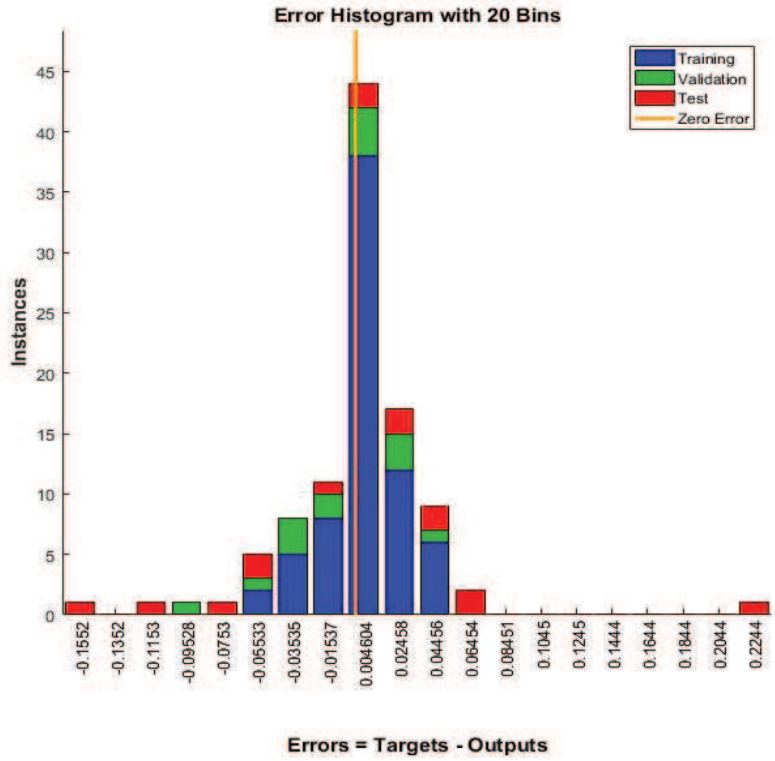


Figure 3.8: PAR error histogram with 20 bins.

Figure 3.8 and Figure 3.10 show that majority of the errors of PAR and BI is close to zero. Comparing Figure 3.9 and Figure 3.11, one can see that the PARś regression is as high as 99.9% for testing. However, the R values of BI are not as good as PARś, but still reasonable. Note that in Regression figures the dot line "Y = T" represents the $R = 1$ where predictions equals targets.

The structure of the NN for predicting energy consumption at community level is shown in

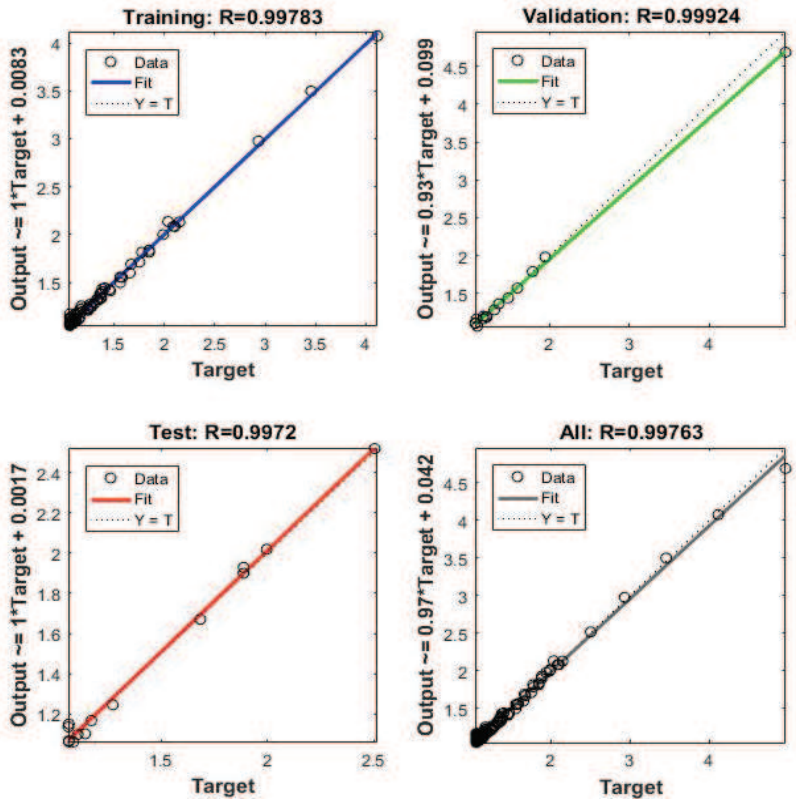


Figure 3.9: PAR regressions of training, validation, and test set.

Figure 3.12. Each input consists of three elements $G(t)$, $E(t-24)$, and $RE(t)$. The output is $Y(t)$, an estimation of our target $Y'(t)$.

Data Pre-processing

Initially, one uses raw data in the training of NN without any data pre-processing.

The results are not as good as expected, as shown in Figure 3.13 and Figure 3.14. One can see that targets are not closely aligned with our prediction. The distribution is quite spread out, and large number of outliers can be observed. We believe there is a way to further increase the accuracy.

As shown in Table 3.1, the data are not well formatted. Values in different features are not at the same scale. For example, the "History" has a broader range, thus the loss function is dominated by it. Moreover, the original range for the target values under the label of "future" is small, making

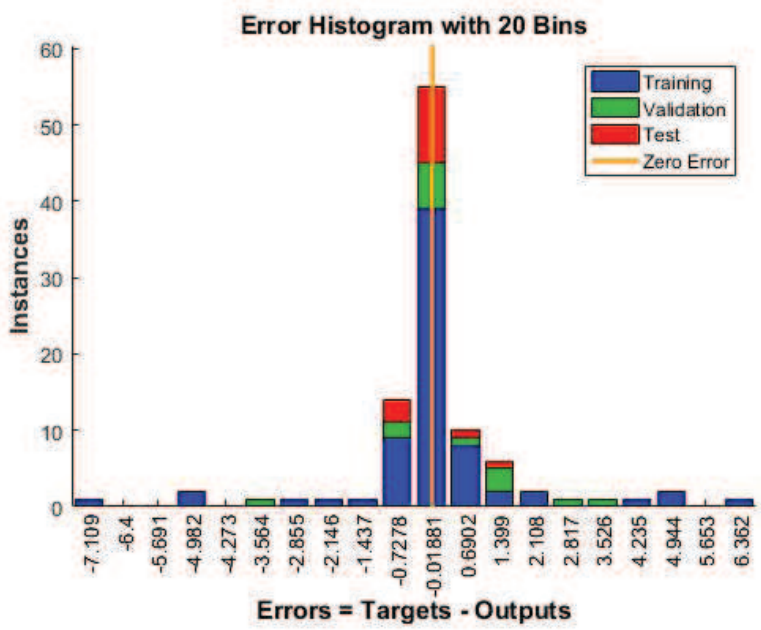


Figure 3.10: BI error histogram with 20 bins.

Table 3.1: A comparison of before applying data pre-processing and after.

Lables	**Max**	**Min**	**New Max**	**New Min**	**New Mean**
Guideline	200e-3	1.00e-3	1	0	6.22e-1
History	1.65e+2	6.55e+1	1	0	6.15e-1
Renewable	4.03e+1	9.83e-3	1	0	4.95e-1
Future	2.00e-3	1.00e-3	1	0	6.26e-1

the optimization sensitive and could stop the training before the network is sufficiently trained.

To solve the problem, data pre-processing is applied. We normalize and re-center it according to the Equation 3.11. The resulting data distributions are in the range of 0 to 1, and centered around 0.5.

The performance of our network is significantly better in terms of both error and regression after applying data pre-processing, shown in Figure 3.15 and Figure 3.16. The distribution of error is more concentrated. The number of large errors is reduced. MSE is 1/5 of the previous one when scaled back to the original scale, and the accuracy has increased significantly to around 99%.

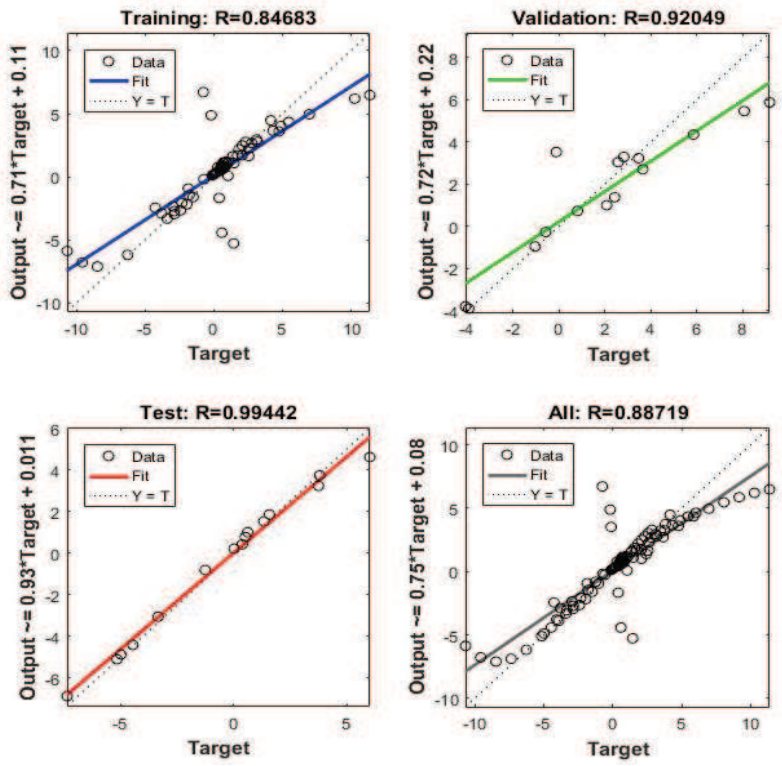


Figure 3.11: BI regressions of training, validation, and test set.

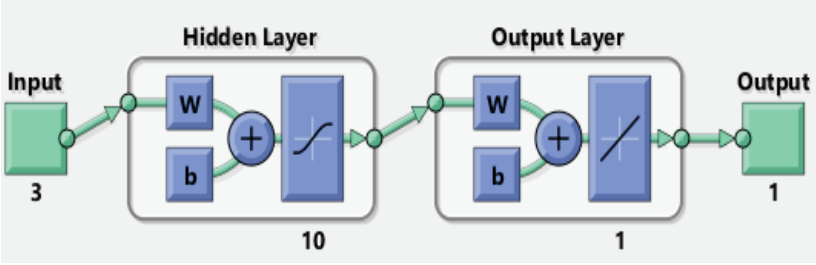


Figure 3.12: Configuration of traditional NN at community level.

Before applying data pre-processing, features are at different scale and their distributions are also different. After applying data pre-processing, inputs and outputs are either compressed or extended to the same scale and shifted to around 0.5.

Comparison of Different Number of Neurons

Deciding how many neurons to be placed can be very tricky. Sometimes, increasing the number

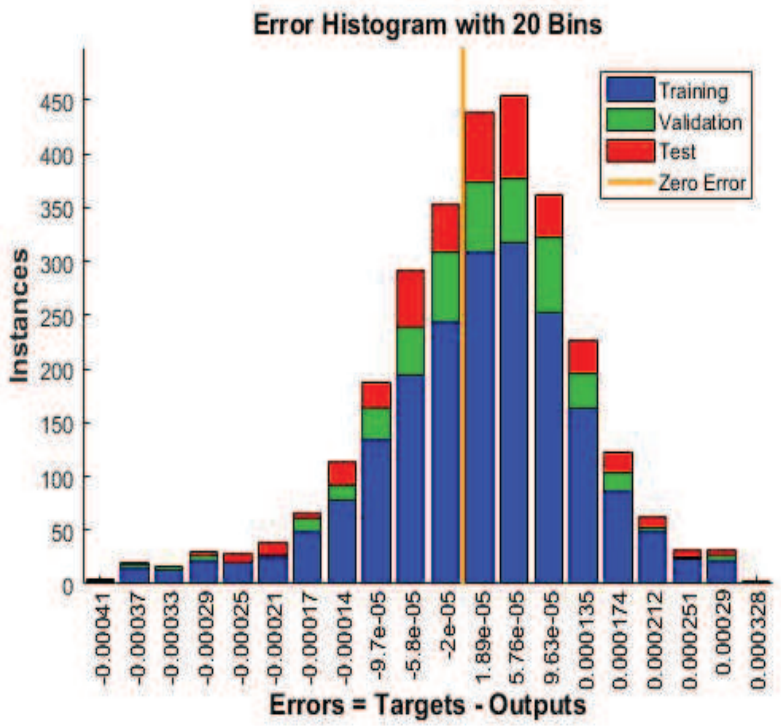


Figure 3.13: Error histogram of traditional NN on raw data.

Table 3.2: Performance with different number of neurons.

Number of Neurons	Mean Square Error	Regression of Test Set
1	1.41635e-3	9.88852e-1
10	1.30805e-3	9.91493e-1
20	1.24818e-3	9.89731e-1
40	1.39445e-3	9.89025e-1
80	1.27178e-3	9.89653e-1

can be helpful, but too many neurons can result in overfitting. Therefore a NN with only one hidden neuron is designed. Surprisingly, the results are very good. Comparing Figure 3.16 and Figure 3.17, one can find that the accuracy only drops a little from 10 neurons to 1 neuron. To further investigate the contribution of the complexity to the performance, one conducts a series of experiments on varying the number of neurons. The results are shown in Table 3.2.

We find that small number of neurons is sufficient to provide accurate prediction of the next dayś energy consumption. Increasing the number of neurons can initially help a little bit in re-

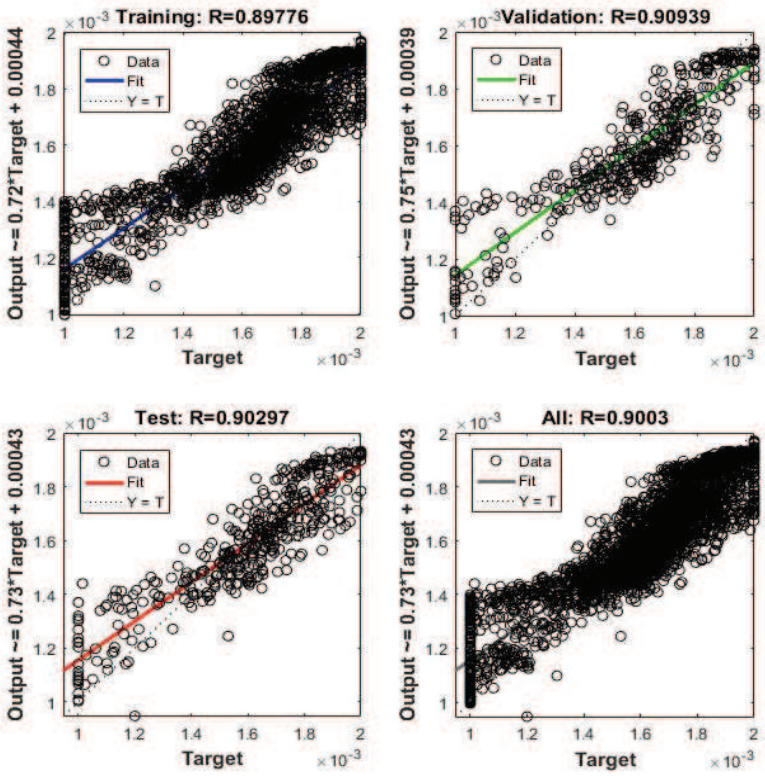


Figure 3.14: Regressions of traditional NN on raw data.

ducing MSE and increasing accuracy. However, the improvement is limited. The reason is that the accuracy is already high, or there is not much useful information can be extracted since the dimension of input and output is low.

Deep NN

Inspired by the promising performance of DNN [47], we build DNN structures and compare their performance. Here, Googleś latest Python library Tensorflow is used to build deep structures. Thirteen different configurations are compared with different training steps as shown in Table 3.3.

The first column in Table 3.3 is the configuration, where each entry represents the number of neurons in corresponding hidden layers. For example, in the first row, it shows a design of 9 layers deep structure. There are 128 neurons in the first hidden layer, 64 neurons in the second, 32 in the third, and so on. And accordingly, there are 128×3 weights and 128 biases assigned to the first hidden layer. The weighted sum is then fed into a rectified linear unit (ReLU) function, defined as

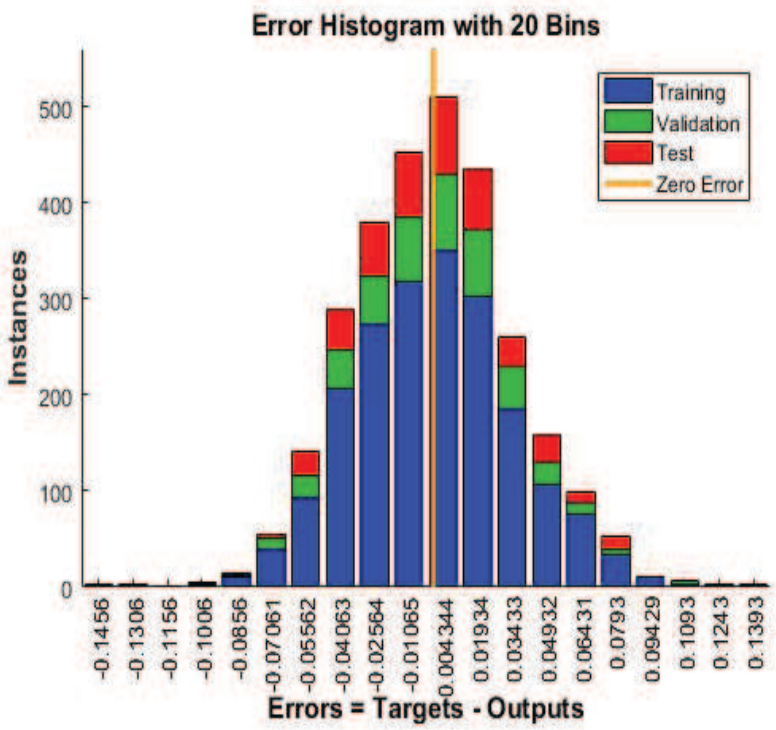


Figure 3.15: Error histogram of NN after data pre-processing.

following:

$$f(x) = max(0, x). \tag{3.13}$$

The major reason of employing ReLU function instead of Softmax or tanh function is the phenomenon called gradient diminishing. In DNN, the gradient backpropagation has multiple stages, and gradients gradually vanish as the structure becomes deeper. ReLU is considered as the most advanced activation function in machine learning [47]. The "loss" column represents the MSE on test sets after training. The "steps" are the number of training iterations. The "global_steps/sec" measures the time elapse of each iteration on the PC that runs the code.

Through comparing the results from the listed 13 configurations, we have preliminarily the following observations.

- Complicated structures do not help much. A structure of 9 layers consists of in total 254

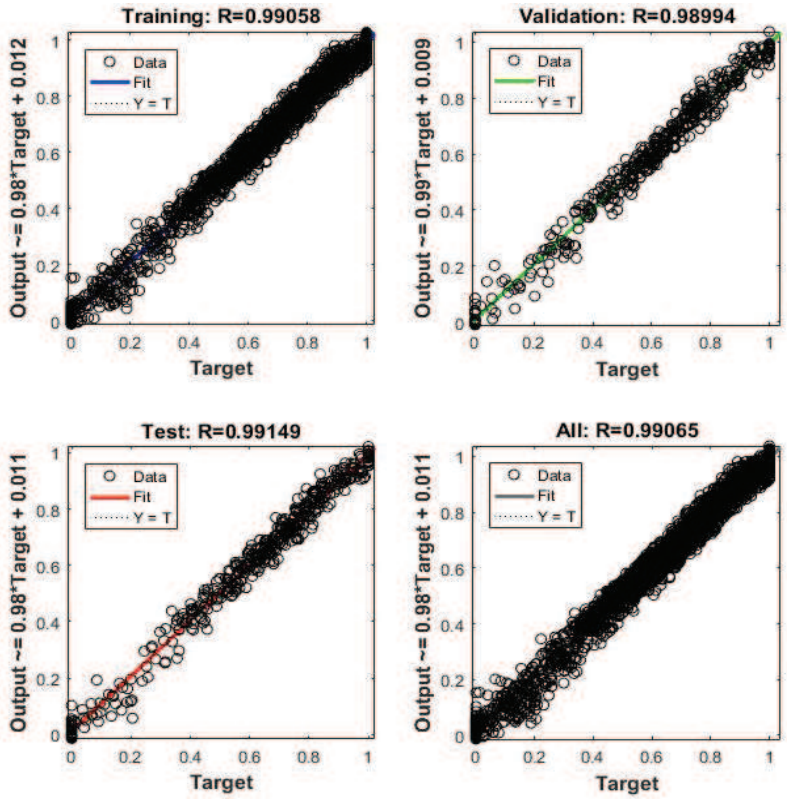


Figure 3.16: Regressions of training, validation, and test set after data preprocessing.

neurons is but no better than a single layer 10 neurons network in terms of MSE and R.

- Increased complexity structure is computationally expensive to implement as shown from global_step/sec.

SWNN

Finally, performance is pushed to the limit through exploring the temporal correlation. In previous designs, all data points are treated as independent. A traditional NN is revised by including previous target values as elements of input for each prediction of $Y(t)$.

At a given time t, target values of previous time $t-1, t-2, ..., t-N$ are fed into the network as elements of input. The method is called "Sliding" because the window has to "slide" forward as the prediction moves on to the next point in sequence. The process is formulated by Equation 11. Same procedures are applied to all t in the sequence recurrently, with the output being depended

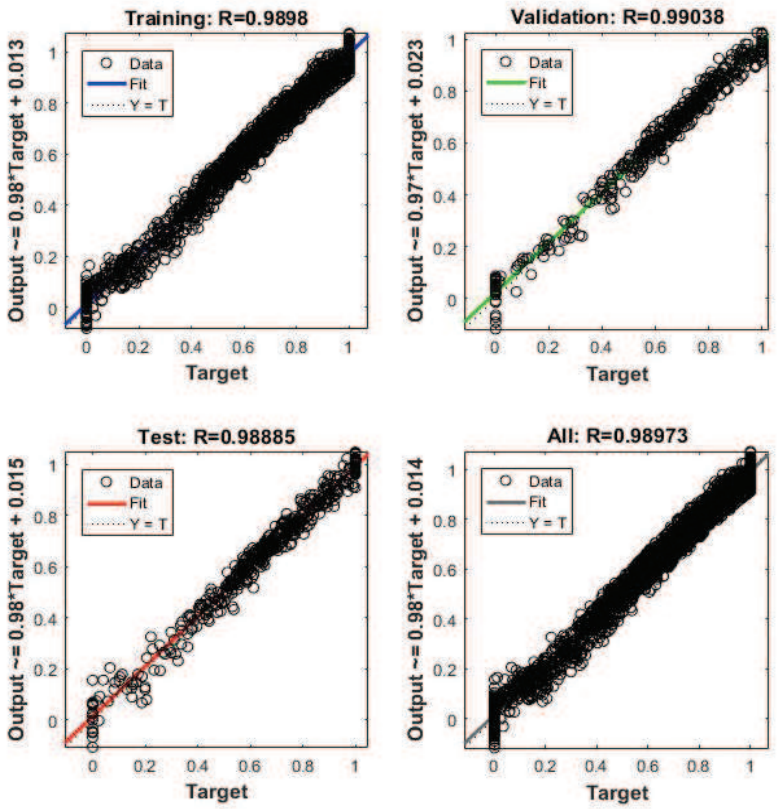


Figure 3.17: Regressions with one hidden neuron NN.

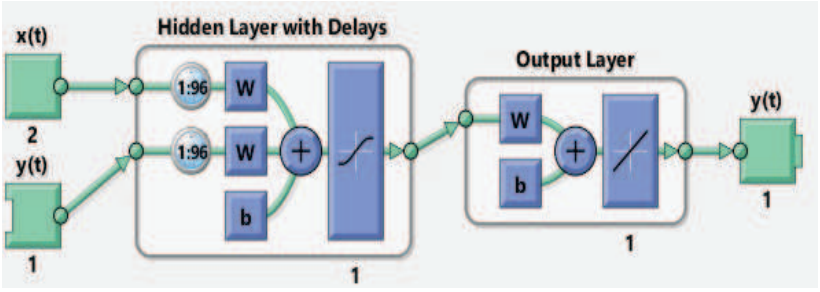


Figure 3.18: Diagram of the SWNN.

on the previous computations.

In our design, one takes $N = 96$, which represents the size of 4 days of past history. With only one hidden neuron in the structure. The result is very inspiring.

From Table 3.4, one can find that MSE is one magnitude lower than previous best performance.

Table 3.3: Comparison of different configurations of DNN.

Configure	Loss	Steps	Global_step/sec	Activation_fn
[128,64,32,16,8,4,2]	0.004334	10000	84.6259	tf.nn.relu
[128,64,32,16,8,4,2]	0.004414	5000	84.3005	tf.nn.relu
[64,32,16,8,4,2]	0.069355	5000	182.838	tf.nn.relu
[32,16,8,4,2]	0.069355	5000	392.62	tf.nn.relu
[16,8,4,2]	0.082669	5000	607.251	tf.nn.relu
[8,4,2]	0.069355	5000	914.194	tf.nn.relu
[4,2]	0.0047	5000	1279.86	tf.nn.relu
[2]	0.005753	5000	1599.83	tf.nn.relu
[10,10,10]	0.004509	5000	639.948	tf.nn.relu
[10,10]	0.004495	5000	743.406	tf.nn.relu
[10]	0.004415	5000	1066.56	tf.nn.relu
[20,20,20]	0.004532	5000	405.095	tf.nn.relu
[20,20]	0.004501	5000	581.758	tf.nn.relu
[20]	0.004637	5000	914.184	tf.nn.relu

Table 3.4: Results of SWNN with one neuron.

Data set	Mean Square Error	Regression of Test set
Training	5.47041e-4	9.95892e-1
Validation	6.64656e-4	9.94738e-1
Test	7.00224e-4	9.94634e-1

The distribution of the errors has a very concentrated shape where the majority of the errors is close to zero. And training, test, and validation sets all have above 99.4% of accuracy, highest among all tested NN structures.

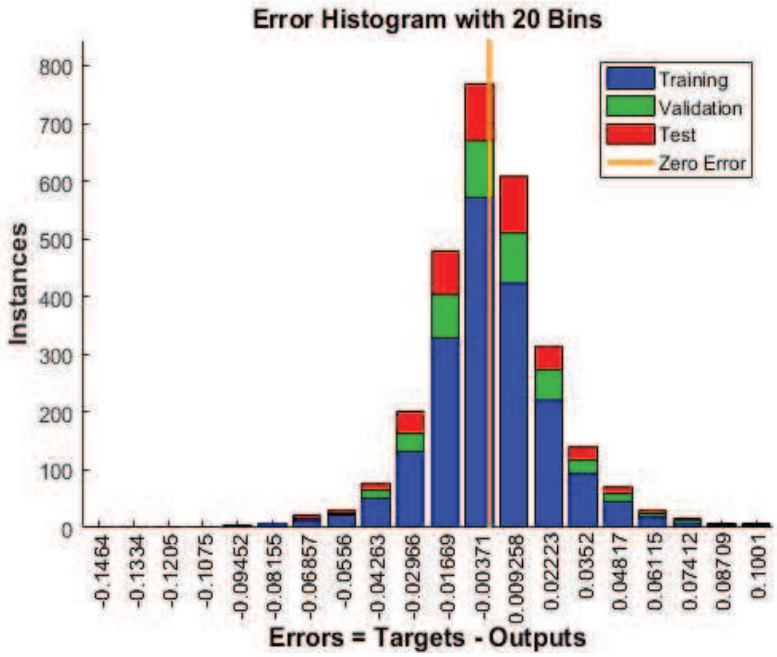


Figure 3.19: Error histogram of SWNN structure with 20 bins.

3.4 Conclusion

Prediction of energy consumption has long been a difficult task for the power industry. Different of types of Neuron Networks are built and compared to find which structure gives best prediction. Data are collected from a community in Belgian, where renewable energy sources (e.g., wind energy source) contribute a significant part of the local energy supply. Single layer NN is used which is proved to be effective in projecting key features of future trend. However, they are not very effective in predicting community level of energy consumption. The quality of data plays an important role in the final performance, so data pre-processing is conducted to project values of different features into the same scale of 0 to 1, and re-center the data around 0.5. The resulting

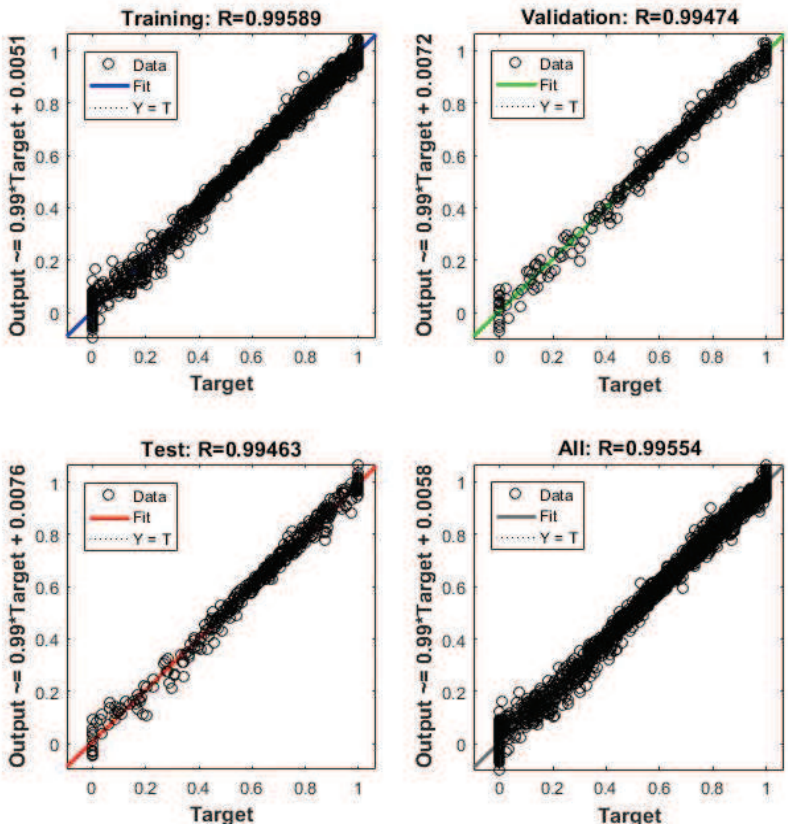


Figure 3.20: Regressions of training, validation, and test set after of SWNN structure. SWNN successfully push the performance of the prediction accuracy to above 99.4%.

accuracy is around 98.9% to 99.0%. We attempt to improve the accuracy further by increasing number of layers and neurons, hoping to add computation capacity and help to extract more underlying patterns. Thirteen different deep structures are built with multiple layers and large number of neurons. Despite significant amount of computation, the technique does not give better results. Lastly, SWNN structure is used to explore temporal correlations in a time series. Target values of previous time in sequence combined with other contributing factors, like availability of renewable energy and guideline price, are used as elements of the input. Most recent 96 hours of data are considered for each prediction, and successfully push the accuracy over 99.4% and reduced MSE by one magnitude.

4. REINFORCEMENT LEARNING BASED IRRIGATION CONTROL [1]

4.1 Introduction

Irrigation management plays a critical role in determining crop yield and water use efficiency. Crop yield largely depends on sufficient water supply. Yet, fresh water resource is limited. Ideally, one likes to irrigate the exact amount of water that is needed by crop, no more and no less. In history, such precise irrigation control was very difficult, if not impossible. Nowadays, wireless sensors, internet and advanced irrigation machines enable Site-Specific Variable Rate Irrigation (SS-VRI) and make the goal of precise irrigation control realistic.

Wireless sensors can help monitor soil moisture levels at real-time and therefore can provide closed-loop feedback to irrigation control. They have caught a lot of attention in irrigation applications. However, many existing works are restricted to the construction of wireless sensor network and demonstration of its benefit. There is some effort on developing advanced irrigation algorithms making use of wireless sensors and/or weather information. A model predictive control approach is proposed in [31]. During a crop growing season, the control takes current sensor and weather data as inputs and makes irrigation decisions according to predicted outcome. Although the use of sensor and weather data can largely overcome the deficiencies of traditional irrigation approaches, the prediction still relies on accurate models, which are not always available. A neuro-dynamic programming method is described in [32]. It is essentially a Markov decision process with model based reinforcement learning. Its drawback is the adoption of a linear model, which is an over-simplification of reality.

We propose a model-free reinforcement learning [54] approach for irrigation control that makes use of soil moisture sensor and weather information. Since it is model-free, it avoids the dependence on potentially inaccurate models. Its decision policy can be learned from simulations as well as real data, including soil moisture sensor data and actual crop yield. In reinforcement learn-

[1] Part of this section is reprinted with permission from "Reinforcement Learning Control for Water- Efficient Agricultural Irrigation" by L.Sun, Y.Yang, J.Hu, D.Porter, T.Marek, and C.Hillyer, 16th IEEE International Conference on Ubiquitous Computing and Communications (UICC), ©2017 IEEE.

ing, a key element is the reward function, which tells if an action is generally good or poor. For agricultural irrigation, the critical reward – crop yield, is not known until the end of a crop season. Such delayed reward is naturally handled by the temporal difference [54] approach in reinforcement learning. Due to limited real data, offline learning through simulation is still important. To this end, fast models based on neural network are developed to facilitate scalable learning. The proposed method is simulated by a fast model developed upon DSSAT (The Decision Support System for Agrotechnology Transfer) [18], which is the de facto standard model for crop growth. The simulation-based comparisons with other methods indicate that our approach can improve net return by around 50% with consideration of both water use and crop yield.

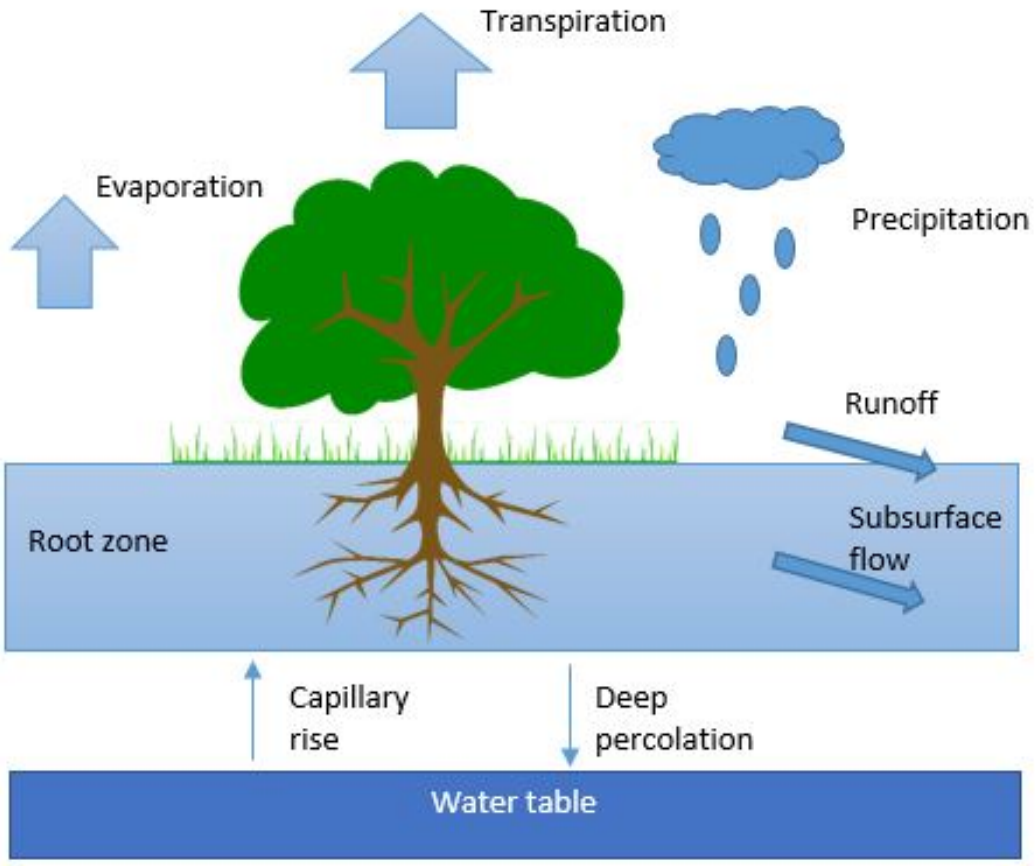


Figure 4.1: Factors of water gain and loss in soil.

The amount of water in soil varies over time depending on many factors. Shown in Figure 4.1, there are 5 major ways of water loss and 2 ways of water gain. Due to gravity(drainage), some of the retained water is pulled out of root zone to deeper layers; such loss is called deep percolation. Evapotranspiration (ET) is the combination of transpiration and evaporation. Evapotranspiration accounts for plant water use (transpiration) and water evaporated from the soil and wet surfaces (evaporation). In arid and semi-arid areas, insufficient availability of water in the crop root zone often is the primary limiting factor for crop yield. Even in relatively humid environments, seasonal or occasional drought conditions can result in water-limiting growing conditions, warranting irrigation as a risk management option to protect against yield loss.

To show the importance of irrigation, DSSAT simulations of maize growth at Temple, Texas according to the weather of 1984 are performed for different irrigation plans. This simple experiment compares yield under no irrigation, and irrigations of 20mm/10days, 30mm/10days, and 40mm/10days. The precipitation during the period is plotted in Figure 4.2. The variation of total water in the profile is depicted in Figure 4.3. The final results are compared in Table 4.1.

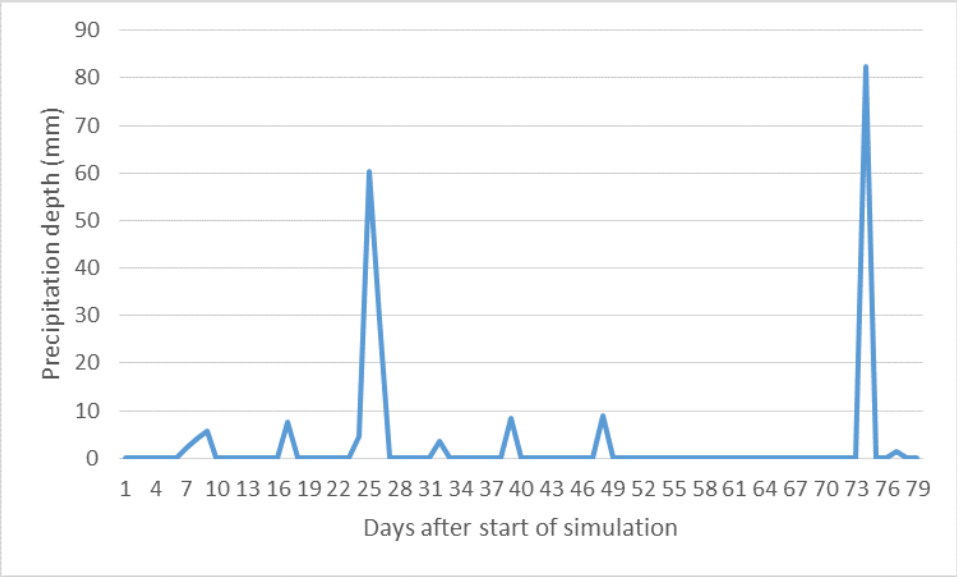


Figure 4.2: Precipitation during the period of simulated crop season. In total, there is 219 mm of rainfall.

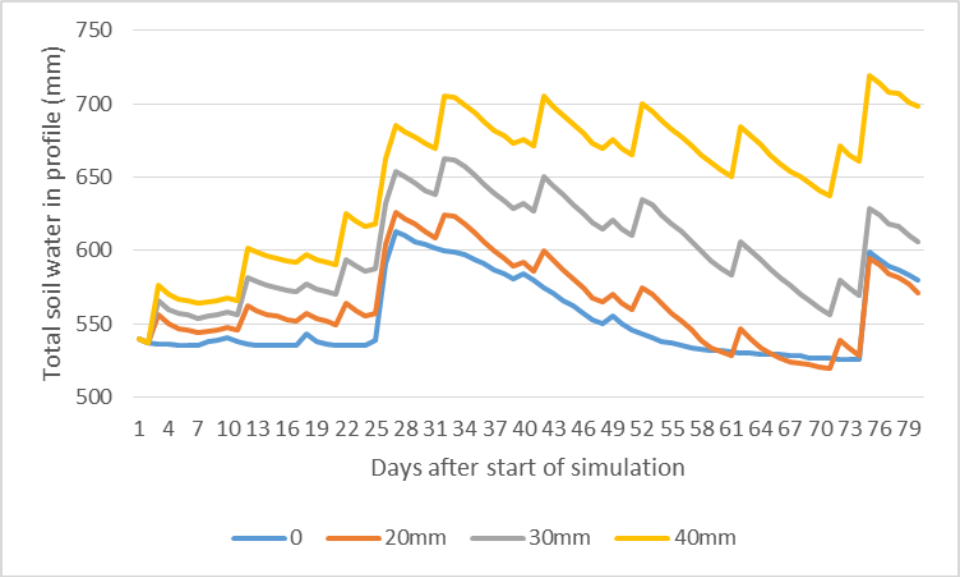


Figure 4.3: Total soil water in the profile during period of simulated crop season. The peak at day 26 is caused by the storm on day 25. Without sufficient amount of water supply, the total soil water drops bellow 540mm for 21 days until the every end of the crop season when it rains on day 74.

Table 4.1: Comparison of yield under different irrigation plan

Irrigation Plan (mm/10days)	0	20	30	40
Yield (kg/ha)	2262	6257	6880	6305

From Table 4.1, one can easily see that irrigation makes a big difference. Under the 20mm/10days plan, with 160mm water supplied in total, yield is increased by 176.6%. But, does more irrigation necessarily increase yield? Not exactly. Comparing the yield under 20, 30, 40 mm/day, one can see that 10mm more water on top of 20mm/10days can boost the yield by 10%, but further increase water supply to 40 mm every time actually suppresses the yield by 8%. Water requirements vary by crop and growth stage. For example, excess water at germination stage can cause poor aeration and discourage proper root development. Therefore, supplying the crop with the right amount

of water at the right time becomes increasingly important (as well as practically achievable) in modern agriculture.

The best timing and amount are very difficult to determine in traditional practice, wherein farmers often use fixed interval irrigation scheduling. The major problems with this strategy are that it lacks flexibility and precision to adjust for precipitation and soil water balance, increasing risk of over-watering (wasting water) or under-watering the crop, leading to yield losses.

Facilitated by the development of wireless sensor technology, threshold-based irrigation scheduling becomes a straightforward approach. Once the soil moisture value drops below a threshold, the system starts to irrigate with a fixed amount. It can largely reduce water waste. Yet, the question of how much to be irrigated is not well answered.

4.2 Related Previous Work

A basic and naïve irrigation decision process is to irrigate fixed amount of water at a constant frequency. Sophisticated irrigation management technologies evolve along two orthogonal directions: advanced algorithms and utilization of modern hardware technologies such as wireless sensors and internet.

The research on algorithmic approaches started decades ago [55], and various optimization techniques have been studied. Irrigation scheduling is formulated and solved by nonlinear programming in [56]. Its weakness is the lack of consideration of many uncertainties in soil, crop and weather conditions. The uncertainties were addressed by stochastic dynamic programming in [57, 58]. Later, genetic algorithm was applied to optimize irrigation scheduling [59]. A learning control technique was proposed in [60], where an analytical model was applied to control soil moisture level without considering crop yield. These techniques were mostly developed before wireless sensor technology, and therefore highly depended on the accuracy of models, which was not always reliable.

Wireless sensor is a critical step in the progress of irrigation technology, as it provides real-time feedback of soil moisture levels. Early works [61, 62] were mostly to demonstrate the benefit of using soil moisture sensors integrated with wireless communication. Soil moisture sensors were

also applied to train neural network-based soil moisture model [63], which was very useful in irrigation scheduling. Infrared temperature sensors and multiband radiometers were installed on the irrigation machine to monitor crop leaf stress level [64], and operated together with threshold based irrigation control [65].

Through internet connection, irrigation control can further incorporate weather conditions and weather forecast information. A model predictive control (MPC) approach was developed [31] that considered both sensor and weather data. Such approach was much more advanced than the offline optimization techniques [56, 57, 58, 59]. However, the prediction part still largely relied on model accuracy. A neuro-dynamic programming-based irrigation control method was proposed in [32]. Its key elements included Markov decision process and model-based reinforcement learning. However, it used a linear model, which was a simplification of the actually complicated crop growth mechanism.

More detailed reviews on irrigation scheduling and control techniques were presented in [66, 67]. Modern irrigation machines allow the application rate of individual nozzles to be separately controlled, and therefore enable the concept of site-specific variable rate irrigation (SS-VRI). However, a study [68] showed that the actual adoption of SS-VRI by farmers is quite limited. A key reason is that there is no mature irrigation control software fulfilling the potentials promised by SS-VRI.

4.3 Problem Model and Formulation

A crops growth process can be viewed as a Markov chain, which is a model for probabilistic transitions among a set of states $S = \{s_1, s_2, ...\}$ over time. In agricultural irrigation, each state $s_i \in S$ is defined as the total soil water (TSW) level at a certain crop growth stage. Depending on specific types of crop and soil, the minimum soil moisture level (management allowable depletion threshold) is set to be higher than the permanent wilting point (PWP), at which plants can no longer extract water from soil and thus die. We set each time step to be 3 days, considering normal irrigation cycle time of most center-pivot machines and speed of water penetration. This setting is tunable. The irrigation decision is to choose among a set of actions $A = \{a_1, a_2, ...\}$ defined

as irrigating the soil until a designated target filling point (TPF) is reached. The decision of what action to take directly affects the state that can be reached at next time step. Therefore, this is a Markov decision process (MDP). In MDP, an action $a_j \in A$ at certain state $s_i \in S$ leads to an immediate reward $r(s_i, a_j)$. The strategy of choosing actions at each time step is summarized as a policy, which usually aims to maximize a long term return or cumulative reward. In agricultural irrigation, the long term return is defined as the Net Return:

$$NetReturn = Y * P_y - C * P_c \tag{4.1}$$

where Y is crop yield with unit kg/ha, C is water use with unit $ha - mm/ha$, P_y and P_c represent product and water price with units of $dollars/kg$ and $dollars/mm$, respectively.

Different from most previous works, the objective of this research is not optimizing merely water use or yield, rather the optimization of net return. Farmers can use it as a direct measurement of their economic gain, and are able to adjust the strategy according to the future market and the water price [69, 70, 71].

4.4 Algorithm Design and Implementation

There are several methods to derive a policy for MDP [54]. Classical MDP assumes complete knowledge of state transition probabilities, which are difficult to obtain in reality. The model-based reinforcement learning approach in [32] assumes a linear model, which is an over-simplification of reality. In this work, we adopt model-free reinforcement learning [54]. It does not depend on any assumption or prior knowledge, but largely acquires experience by interacting with the environment.

At each time step, the agent, which in our case is the irrigation controller, depending on current state $s \in S$, takes action $a \in A$, and observes an immediate reward r. At next time step, the previous reward r is discounted by the factor γ, whose value is between 0 and 1. This term γ controls the preference of the agent's behavior. When $\gamma = 1$, the agent would take long-term strategy, while if 0, the agent would only strive for large immediate reward. The quality of a state-

action pair is specified by function $Q(s, a)$, which defines the expected cumulative reward by being at state s and taking action a. Its value is largely decided by reward resulting from a trajectory of state-action pairs. A challenge in irrigation is that crop yield, the critical reward, is not known until the end of crop season. The temporal difference learning algorithm SARSA(λ) [54] is an approach for handling such delayed reward. It updates Q functions for state-action pairs backwards according to the eligibility trace, in which λ controls the eligibility (or the relevance of later reward to previous state-action pairs).

A common issue faced by reinforcement learning is the tradeoff between exploitation and exploration. To be more specific, an agent explores the environment and learns from experience. However, at early stages, heavily relying on learned Q values or exploitation to make decisions may close doors for discovering better routes. While at later stages, it would be a waste to spend too much time on exploration rather than using existing knowledge. A popular approach for addressing the exploitation-exploration tradeoff is the ϵ-greedy algorithm [54], where the action with the so far largest Q value is taken with probability $1 - \epsilon$ and an action is randomly chosen among all actions with probability ϵ. At state s, action a is taken according to ϵ-greedy algorithm. Then reward r is received and the state transits to s'. Let a' be the next action according to ϵ-greedy algorithm. The Q value of the state-action pair (s', a') is updated with the temporal difference:

$$\delta = r + \gamma Q(s', a') - Q(s, a) \tag{4.2}$$

The program keeps a record of the eligibility trace $e(s, a)$. After each visit, the eligibility value of current state action pair is added by 1:

$$e(s, a) \leftarrow e(s, a) + 1 \tag{4.3}$$

After each visit, all entries of the Q table are updated according to the δ and $e(s, a)$ with a learning rate α.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a) \tag{4.4}$$

The eligbility is discounted by the product of γ and λ so that rewards obtained at later time steps are updated according to relevance of previous state-action pair. The longer distance, the smaller relevance, and therefore less weights on the updates.

$$e(s, a) \leftarrow \gamma\lambda e(s, a) \tag{4.5}$$

The above process is repeated until either the Q table is converged or the policy is sufficiently stabilized. Please note this process contains both learning and decision making.

In order to learn what is good and bad in terms of policy, the agent must interact with the environment sufficiently. However, such process is extremely slow since learning from one actual crop season takes 90 -120 days depending on the types of crop and time of planting. Thus, to shorten the learning process, simulation model packages such as DSSAT [18] can be of great help. However, directly incorporating DSSAT is very difficult, since the control of its irrigation scheduling requires either manual editing through its own GUI or a perfect understanding of its source code, which was written in Fortran.

To make the Q function training more scalable, we develop a cascaded Neural Network (NN) model as a surrogate to DSSAT. The front end of this model is an NN that takes irrigation and weather information as inputs and predicts TSW for a geographic location. The backend model is another NN that predicts crop yield given daily TSWs of an entire crop season. Since final yield is closely related to the TSW during the simulated crop season, one can run some random irrigation plans and extract TSW tables as inputs, and use the obtained yields as targets to train the backend crop yield NN. The input to crop yield NN is from the output of front end NN that predicts daily TSW. TSW level can be affected by not only precipitation and irrigation, but also ET, runoff, and percolation, which themselves vary by many factors such as soil type, solar radiation, wind speed, and temperatures, etc. In DSSAT, the TSW is calculated by a set of very complicated but relatively accurate models. In our work, data obtained from DSSAT simulations are used to train the front end TSW NN. Figure 4.4 shows the steps to construct our simulation system.

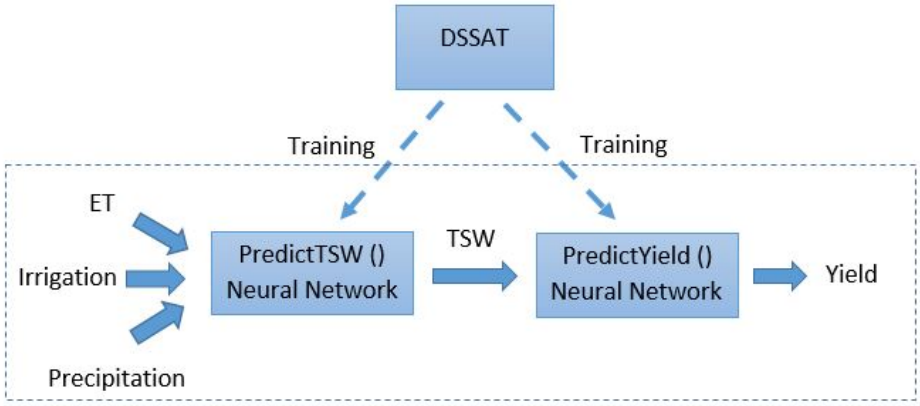


Figure 4.4: The process to construct cascade NNs for simulation using DSSAT data.

Data: time step i, action $a \in A$
Result: Total Soil Water on $j+3$ day and Irrigation amount I_j
$j = (i-1) * 3 + 1$ // j is index of days ;
$I_j \leftarrow IrrigationAmount(TSW_j, TFP_a, Weather)$;
if $I_j < 20$ **then**
  $I_j \leftarrow 0$
end
for $k = j+1$, k++, $k < j+4$ **do**
  $TSW_k \leftarrow PredTSW(TSW_{k-1}, I_{k-1}, ET_{k-1}, R_{k-1})$
end

Algorithm 7: Generate daily TSW and irrigation record

In the SARSA(λ) algorithm, although each time step is 3 days long, precise prediction of yield still requires daily TSW data. As described in Algorithm 1, the program first translates time step i to days $[j, j+2]$. Then, it calculates irrigation water depth I_j for day j according to current TSW_j, and the TFP (Target Filling Point) decided by actions in A, using a function $IrrigationAmount()$. Because frequent small irrigation applications result in large evaporation loss and discourages deep root development, we require the depth of any irrigation application to be at least $20mm$. Once the amount is determined, the program runs NN function $PredTSW()$ to produce TSW values for the following 3 days. The function takes current TSW, irrigation (if any), ET, and precipitation R as inputs.

The reward function design for the SARSA(λ) learning in irrigation warrants particular discus-

sion. The long term return is the net return resulted from crop yield and water expense. Since water use occurs multiple times throughout a crop season, one approach is to count their expense immediately after each irrigation action. However, this approach brings two problems. First, the reward due to water use is negative since obviously it is to be minimized instead of being maximized. A negative reward often results in negative Q values, which cause trouble in the exploitation-exploration tradeoff. More specifically, a good action a^* may temporarily have negative Q value while another under-explored action a' may have zero Q value even though it is a very poor action. Such discrepancy would mislead the subsequent learning process. Second, the reward is discounted by eligibility trace in the SARSA(λ) learning. As such, the same dollar amount for water use and crop yield is treated with different weights. To overcome these problems, we defer the reward associated with water use to the end of crop season. In other words, all time steps in the middle of a season have 0 immediate reward and there is only a single reward - the net return, at the end of a season.

Algorithm 8 provides a complete description of the learning process. Every simulated crop season is considered as an episode, consisting of n time steps. The program starts with a randomly generated Q table, note that Q values of all state-action pairs should be relatively small in magnitude compared to the rewards. At each time step, the agent takes an action and runs function $SimuTSW()$ to obtain TSW and irrigation record for the current time step. The reward function is designed such that all the immediate rewards are 0, except for the last one. The final reward is the net return, calculated by Equation (4.1). Note that if net return is smaller than a desired value $threshold$, then it will be assigned a small negative value -50. Such design helps to differentiate good policy from bad ones dramatically. In addition, a small negative value would stimulate agent to favor other unexplored state action pairs, thus making it faster to find good policies. Q table and eligibility trace are updated at the end of each time step accordingly. Upon completion of each episode, the eligibility is reset to be all 0s, but the Q table will be kept as a reference to continue learning until convergence.

Initialize $Q(s, a)$ arbitrarily;
repeat
  forall $e(s, a)$ **do**
    $e(s, a) \leftarrow 0$ //Initialize eligibility trace
  end
  for $i = 1$, $i{+}{+}$, $i < n + 1$ **do**
    $s \leftarrow State(i, TSW_i)$;
    Take action a;
    $[TSWi + 1, I_i] \leftarrow SimuTSW(i, a)$;
    $s' \leftarrow State(i, TSW_i)$;
    $a' \leftarrow greedy(\epsilon, Q, S_{i+1})$;
    if $i < n$ **then**
      $r \leftarrow 0$
    else
      $W \leftarrow Sum(I) * WaterPrice$;
      $Y \leftarrow PredYield(TSW) * ProductPrice$;
      $NetReturn \leftarrow Y - W$;
      if $NetReturn < threshold$ **then**
        $r \leftarrow -50$
      else
        $r \leftarrow NetReturn$
      end
    end
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$;
    $e(s, a) \leftarrow e(s, a) + 1$;
    forall s *and* a **do**
      $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$;
      $e(s, a) \leftarrow \gamma\lambda e(s, a)$;
    end
    $a \leftarrow a'$;
  end
until $Q(s, a)$ *converges or policy sufficiently stabilized*;

Algorithm 8: SARSA(λ) in Irrigation

Table 4.2: General information of test cases

Location	Temple	Kunnunurra	Hyderabad	Saskatchewan
Soil Type	Clay	Clay	Clay	Loam
Cultivar	Maize	Maize	Maize	Wheat
Planting data	05/12/1984	06/12/1982	05/12/1983	05/25/1975

4.5 Experiment

To evaluate effectiveness of the proposed techniques, simulations are run on 4 different locations, as shown in Table 4.2: Temple, Texas, United States; Kunnunurra, Australia; Hyderabad, India; and Saskatchewan, Canada. The first three fields are planted with maize, and the last field is planted with wheat. These testcases are summarized in Table 4.2.

All NNs for predicting TSW and crop yield are trained with single hidden layer consisting of 10 neurons. The training algorithm is Levenberg-Marquardt [72]. Samples are divided into training, validation, and testing sets. NNs are trained on training sets, and validated by validation sets, which are used to tell when to stop training. The accuracy of NNs are measured on the test sets to provide unbiased results. Figures 4.5 to 4.8 show the training performance for the Temple case. Regression measures the correlation between targets and outputs, the closer to 1, the better. In the error histogram, the smaller and more concentrated around zero, the more accurate. The training performance evaluation for the other 3 cases are not presented here. But the statistics are all very promising as summarized in Table 4.3.

Table 4.3: Summary of NNs training regressions

Regression	Temple	Kunnunurra	Hyderabad	Saskatchewan
Yield	>0.97	>0.98	>0.98	>0.96
TSW	>0.99	>0.99	>0.99	>0.99

Using NN has significantly improved learning efficiency, as running each episode costs less than 2 seconds. If all done in DSSAT, considering manually adjusting irrigation plan takes approx-

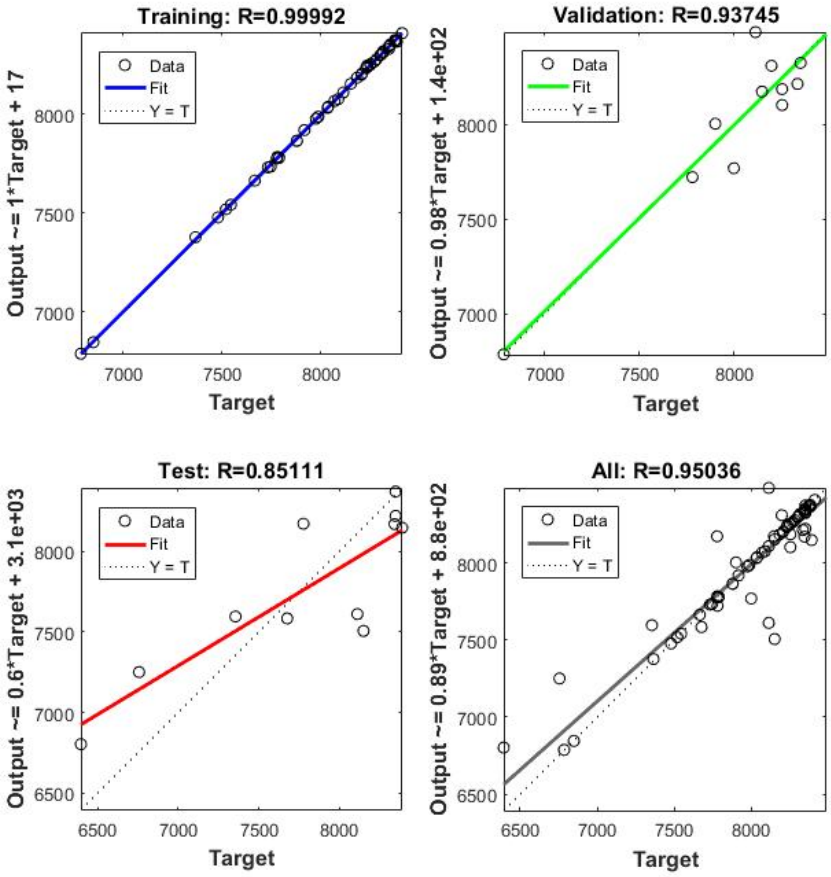


Figure 4.5: NN training regression for predicting yield at Temple, TX, US. The overall regression goodness of fit above 0.95.

imately 1 hour for each episode, and the rule of thumb that Q-table normally converges after 500 iterations, the time cost would be unmanageable.

States in reinforcement learning are defined according to crop growth stage and TSW. The state definitions for Temple and Saskatchewan are shown in Table 4.4 and 4.5, respectively, and the state definitions for the other two locations are similar. In the tables, the header rows are ranges of TSW with unit mm. The header columns are time steps. Each entry in the table is a state ID. Since the soil and crop type in Temple, Kunnunurra, and Hyderabad are the same, the division of TSW levels are the same in those cases. However, as they differ in geolocation, planting date, and weather conditions, the length of the crop seasons and definition of growth stages have some

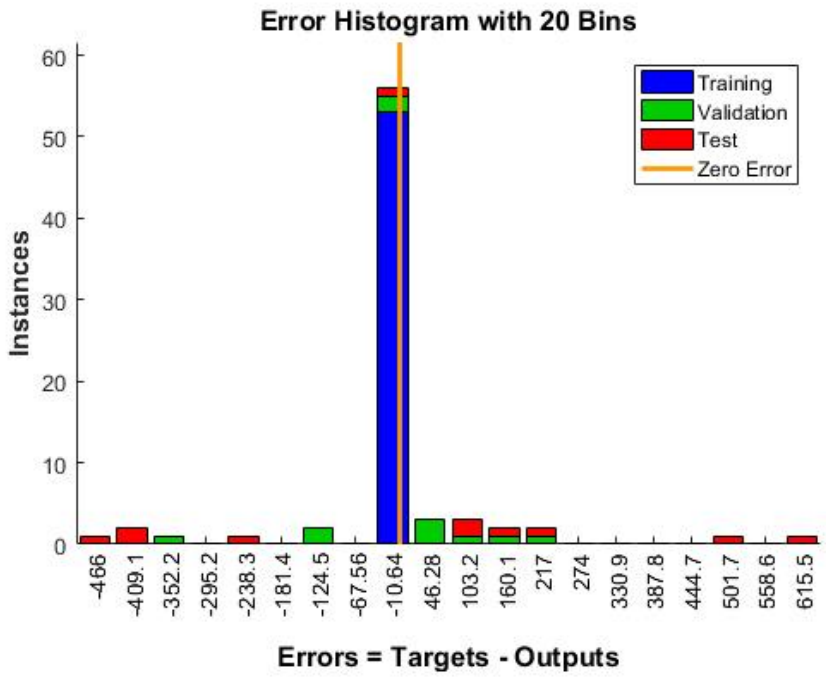


Figure 4.6: NN training error histogram for predicting yield at Temple, US. Most errors are concentrated around 0, the relative error is within 0.1%

variance. The actions for all four cases are defined in Table 4.6. Action 1 is to wait, the rest actions are to irrigate until TSW reaches designated Target Filling Point (TPF).

Table 4.4: State definition of case Temple, Texas, United States

State	<=540	(540,545]	(545,550]	(550,560]	>560
<=7	1	2	3	4	5
<=15	6	7	8	9	10
<=21	11	12	13	14	15
>21	16	17	18	19	20

The proposed learning control method is compared with

- Fixed scheduling: a fixed amount of water is irrigated every 10 days.
- Threshold-based irrigation: a fixed amount of water is irrigated at a time step (3days) if the

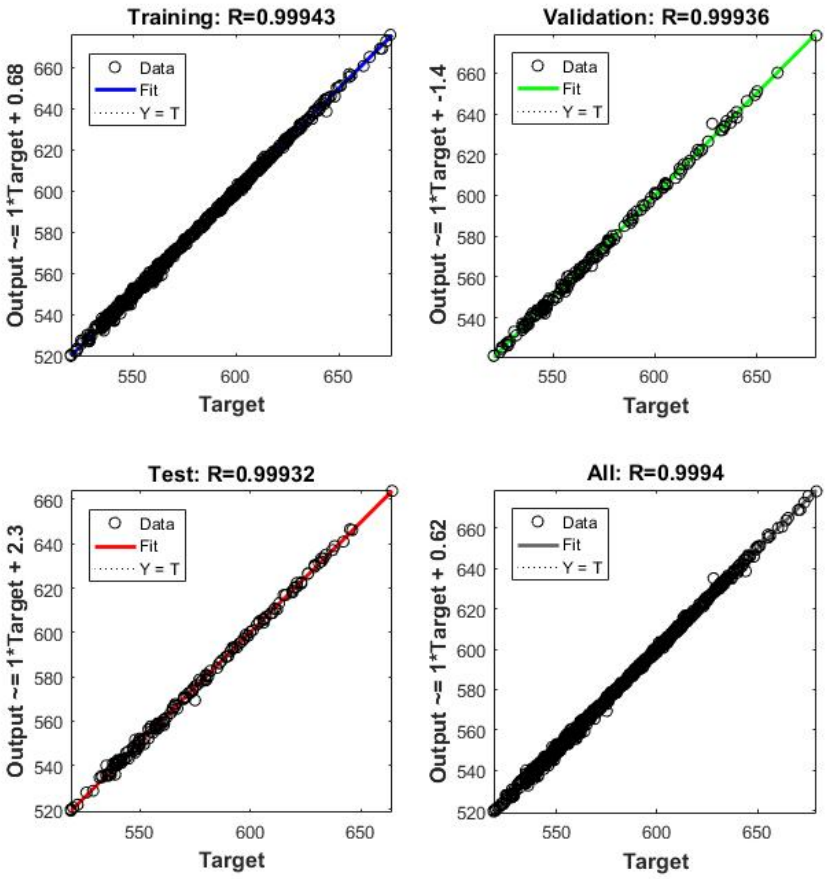


Figure 4.7: NN training regression for predicting TSW at Temple, US. The performance is almost perfect with all sets above 0.999.

Table 4.5: State definition of case Saskatchewan, Canada

State	<=320	(320,325]	(325,330]	(330,340]	>340
<=13	1	2	3	4	5
<=20	6	7	8	9	10
<=27	11	12	13	14	15
>27	16	17	18	19	20

soil moisture level is below a certain threshold.

At the time of this analysis was conducted, according to the Food and Agriculture Organization of the United Nations (FAO), international maize and wheat prices were around 200 USD/tonne[69,

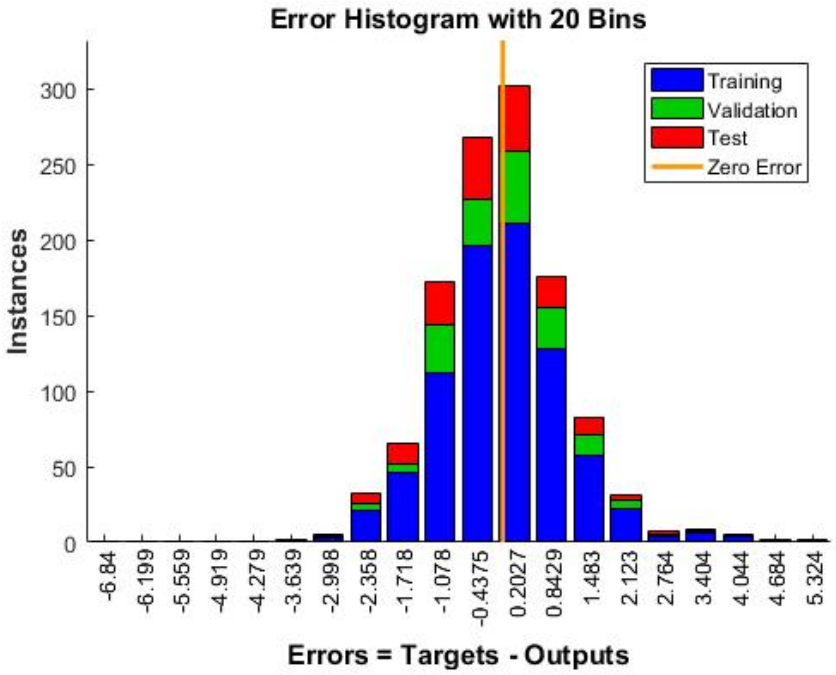


Figure 4.8: NN training error histogram for predicting TSW at Temple, US. Most errors are within +/-2 mm

Table 4.6: Definition of actions in all four cases

	1	2	3	4
Temple	wait	560	570	580
Kunnunurra	wait	560	570	580
Hyderabad	wait	560	560	580
Saskatchewan	wait	340	350	360

70]. The cost of irrigation for every 1ha-mm/ha is about 1 USD[71]. The following results are produced based on those price settings. It is worth mentioning the way we handle the exploration and exploitation trade off. To achieve a good balance, ϵ is initialized to be 0.7 to encourage exploration. As the experience accumulates, the decision making increases its reliance on existing knowledge. As learning continues, ϵ decreases. To be more specific, if the number of learned episode $N < 300$, $\epsilon \leftarrow 0.7 - 0.002N$; when $N >= 300$, $\epsilon \leftarrow \sqrt{1/N}$ for each episode.

The advantage of adopting learning method can be seen from comparing TSW curves in Fig-

Table 4.7: Comparison of performance under different irrigation methods in the Temple case

	Yield (kg/ha)	Total irrigation (ha-mm/ha)	Net Return (dollars/ha)
Learning	9224	236	1608
Threshold 540mm	6576	160	1155
Threshold 550mm	7530	200	1306
Threshold 560mm	7876	240	1335
Fixed 20mm	6993	180	1218
Fixed 30mm	8948	270	1519
Fixed 40mm	8975	360	1435

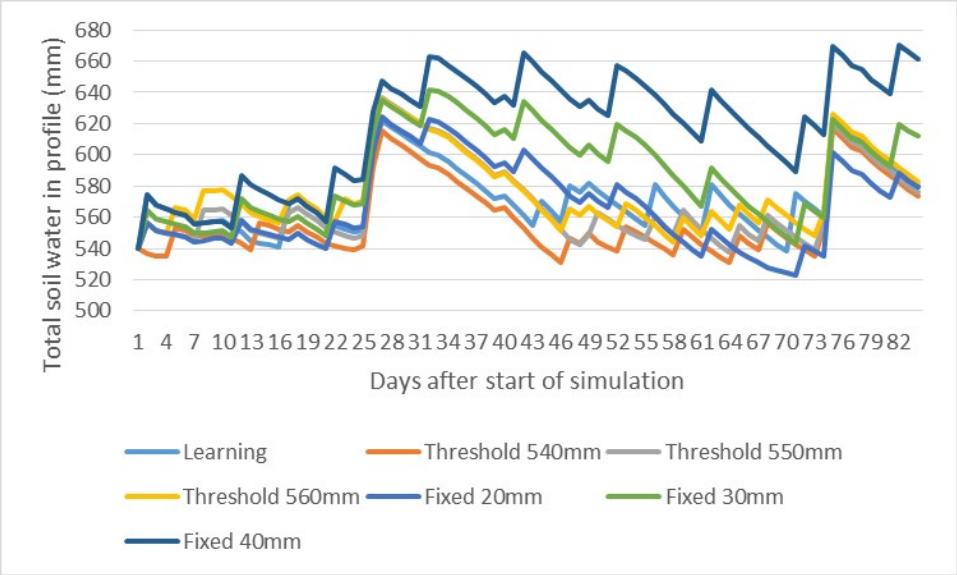


Figure 4.9: Comparison of TSW profile under different irrigation methods in the Temple case.

ure 4.9 to 4.12. In Figure 4.11, the TSW takes off, even the field does not need that much water. Yet, when water is lost too quickly, in Figure 4.12, fixed method could not keep up with crop demand. Although threshold-based method avoids these problems, it is still not flexible enough. In Figure 4.9, the learning method initially keeps TSW at low level but increases water supply at later stages. Such arrangement makes perfect sense in agriculture, since at early growth stages, like germination, crops are small and roots are shallow, thus crop water demand is lower compared to later stages. The computer does not understand the science in agriculture, but somehow learned

Table 4.8: Comparison of performance under different irrigation methods in the Kunnunurra case

	Yield (kg/ha)	Total irrigation (ha-mm/ha)	Net Return (dollars/ha)
Learning	11279	303	1952
Threshold 540mm	6530	280	1026
Threshold 550mm	8732	280	1406
Threshold 560mm	8393	380	1298
Fixed 20mm	3482	200	496
Fixed 30mm	4671	300	634
Fixed 40mm	8743	400	1348

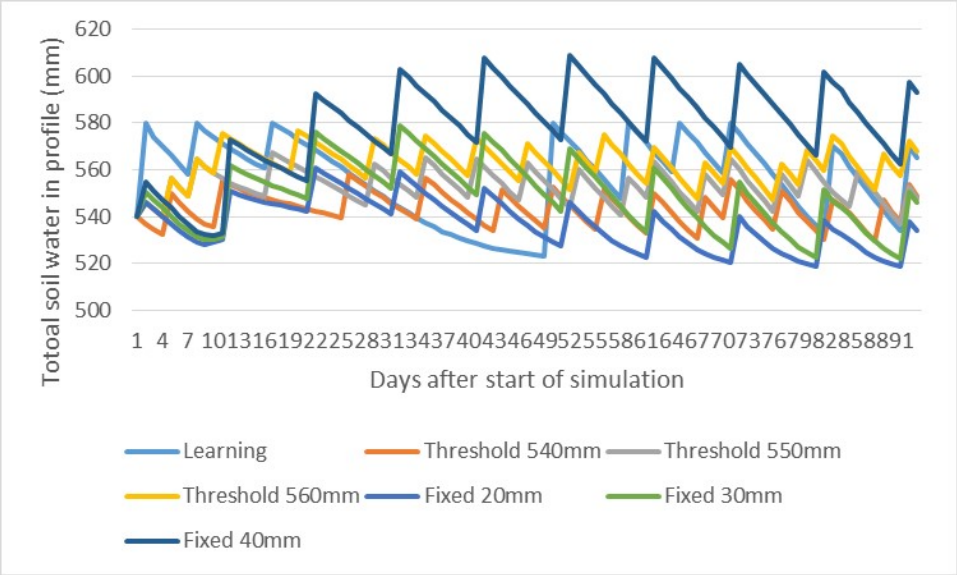


Figure 4.10: Comparison of TSW profile under different irrigation methods in the Kunnunurra case.

the trick from exploration.

From the result shown in Table 4.7 to 4.10, one can see that the performance of the learning method stands out in every case. In Temple, the net return was improved by 27.1% and 15.6%, compared to the average of threshold-based and fixed scheduling method, respectively. In Kunnunurra, the increased profits are 56.9% and 136.2%; in Hyderabad, 96.4% and 38%, and Saskatchewan, 6.4% and 49.2%. Not only that, the learning also conserves water; in most cases,

Table 4.9: Comparison of performance under different irrigation methods in the Hyderabad case

	Yield (kg/ha)	Total irrigation (ha-mm/ha)	Net Return (dollars/ha)
Learning	5493	125	973
Threshold 540mm	2100	80	340
Threshold 550mm	3217	120	523
Threshold 560mm	3815	140	623
Fixed 20mm	4079	160	655
Fixed 30mm	5303	240	820
Fixed 40mm	4790	320	638

Table 4.10: Comparison of performance under different irrigation methods in the Saskatchewan

	Yield (kg/ha)	Total irrigation (ha-mm/ha)	Net Return (dollars/ha)
Learning	8146	210	1419
Threshold 540mm	7377	220	1255
Threshold 550mm	7893	200	1378
Threshold 560mm	8031	240	1366
Fixed 20mm	6478	220	1075
Fixed 30mm	6680	330	1006
Fixed 40mm	6057	440	771

the water consumption under learning method is lower than threshold-based and fixed scheduling methods. Of all tested 4 cases, the learning method outperforms the threshold-based method by 46.7%, and fixed scheduling by 59.8% on average net return.

4.6 Conclusion

Improved management of agricultural irrigation plays a critical role in addressing the challenge of fresh water shortage. The progress of wireless sensor and internet technologies allows advanced site-specific variable rate irrigation, which has not been fully exploited yet. A reinforcement learning based control approach is proposed. Its training can be carried out either through online crop growth season or offline simulations. A neural network infrastructure is built to facilitate efficient trainings. Its successful implementation opens a promising alternative in future computational

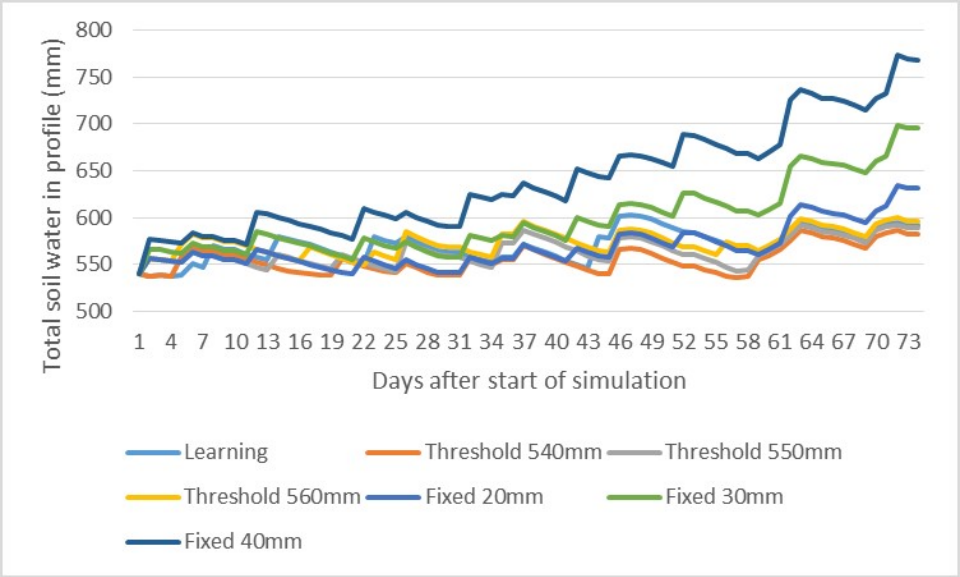


Figure 4.11: Comparison of TSW profile under different irrigation methods in the Hyderabad case.

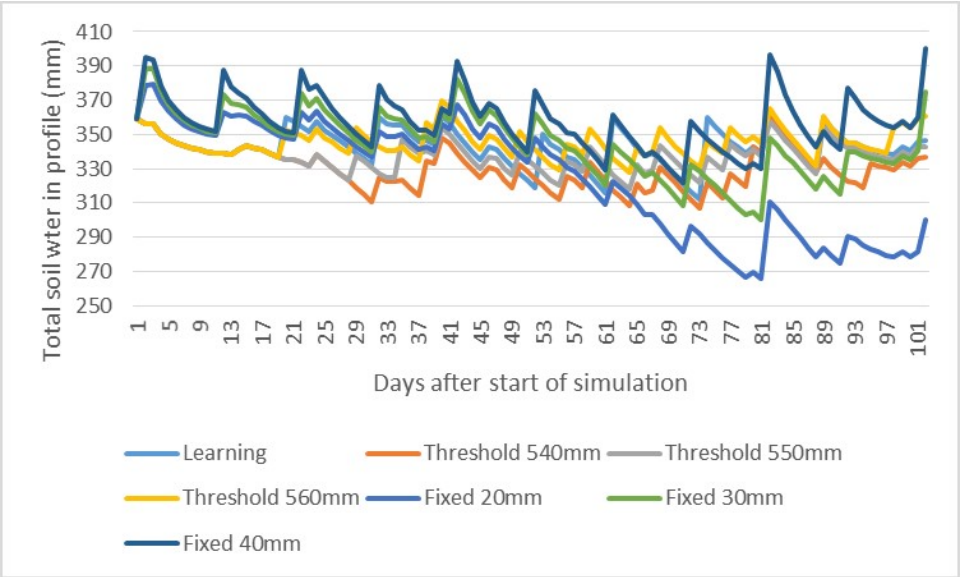


Figure 4.12: Comparison of TSW profile under different irrigation methods in the Saskatchewan case.

agricultural research. Simulations for different crop types at various geographic locations show that the proposed method outperforms fixed irrigation scheduling by 59.8% and threshold based approach by 46.7% on average net return.

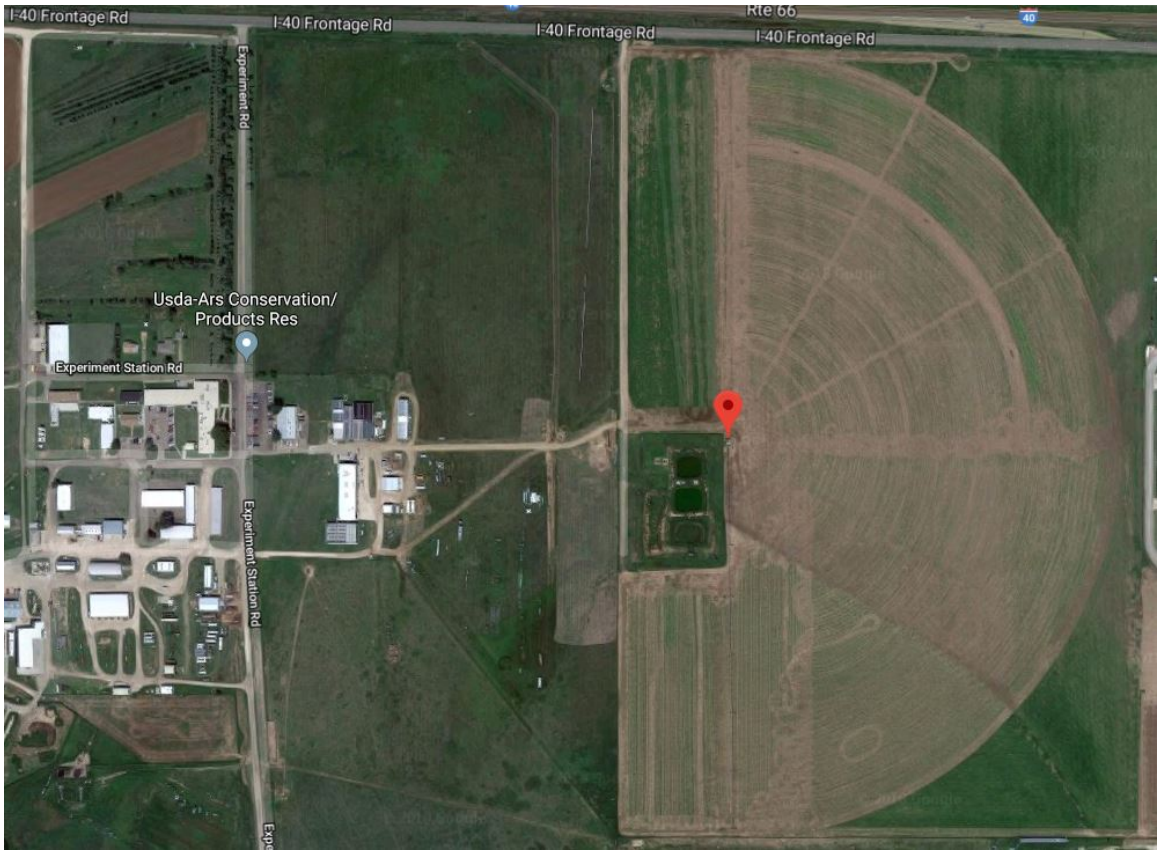


Figure 5.1: Satellite image of the experiment site at Bushland, Texas.

A real operational smart irrigation system was built at United States Department of Agriculture (USDA) Conservation and Production Research Laboratory at Bushland, Texas. Figure 5.1 shows the exact location of the facility. The main purpose of this project was to prove, test, and evaluate the concept of a fully automated irrigation system based on artificial intelligence (AI) and IoT technologies. The system was designed to be highly efficient and reliable. It can collect real-time soil moisture sensor readings through wireless network, download latest weather forecast, determine the best action to take in order to maximize farmers' economic return, and execute

precisely in most terrain and weather conditions. It can be controlled through a local manual control board, or by a web-based user interface. Water is pumped through underground pipeline to the center pivot, figure 5.3 and 5.4. Users can program it to perform certain tasks, monitor its operations, or intervene if deem necessary. Operational data are kept locally on control board, and multiple security layers were added to make sure that only authorized personnel have access.

5.1 Architecture

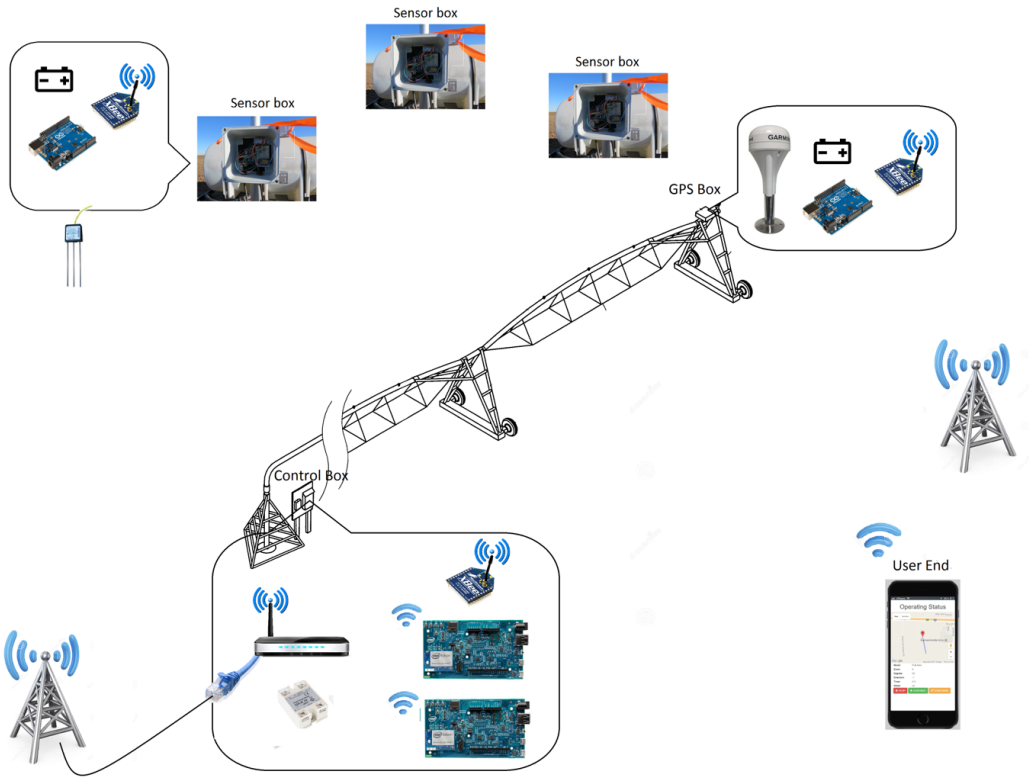


Figure 5.2: System architecture.

Figure 5.2 presents the overall architecture. It consists of six subsystems:

- Control
- Water Supply

- Pivot movement
- GPS and Soil Moisture Sensors
- Communication
- User Interface

Soil moisture readings are collected from sensors and sent through wireless modules to center pivot. At center pivot, two Edison boards sit inside the control box. One of the boards is designated as the "Information Center", whose jobs are collecting soil moisture sensor signals, GPS coordinates, downloading weather data, and storing those data in proper format. The other Edison board is name "Control Center", which analyzes soil moisture and weather data, executes irrigation plan, and communicates its operation to the user(s). The communication system has three data links, figure 5.5 and 5.6, control box to sensors, GPS, and the Internet. An authorized user can choose to either control operations on site by using a manual interface, or use a web based application and realize the control remotely.

Figure 5.3: Water is pumped from a local reservoir through underground pipeline to the pivot. Mr. Marek was controlling the water flow through a value.

Figure 5.4: A digital flow meter is installed and connected to the control center.

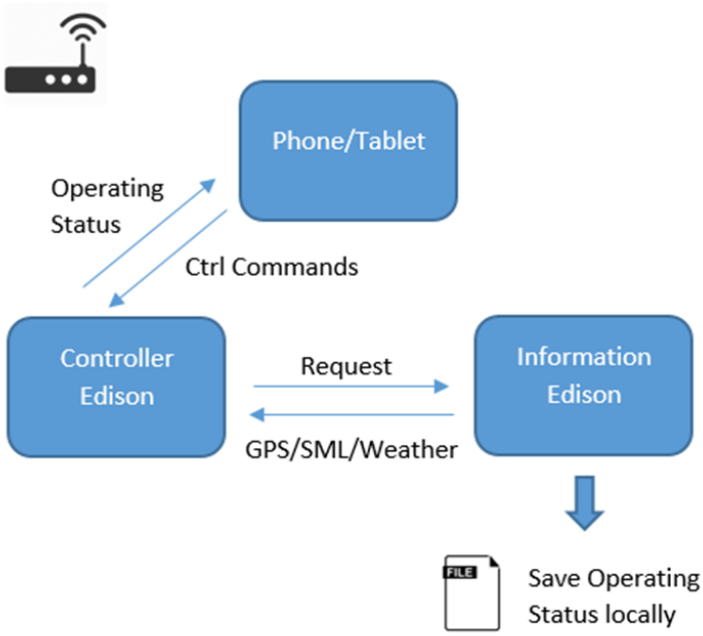


Figure 5.5: Communications between users' mobile terminals, controller board, and information board.

Figure 5.6: An antenna was mounted at pivot, high enough to have clear line of sight to GPS and sensors.

Figure 5.7: The author is installing a sensor box.

At each sensor location, three sensors were embedded to the ground at different depths, and readings from sensors are transmitted back to the control center. Sensors signals are relayed through a wireless module to the center pivot. Both sensors and wireless module are powered by 12V batteries, which can last for 3 months and can be recharged. They are housed in a water proof sensor box above ground. The boxes are mounted high enough to extend transmission distance and prevent signals been blocked by tall crops, as shown on figure 5.7.

5.2 Control System

Prototypes were initially made based on Arduino Uno platform, figure 5.8. They were cheap and simple, yet lack the ability to perform more complex functions. These controllers were later replaced by Edison boards.

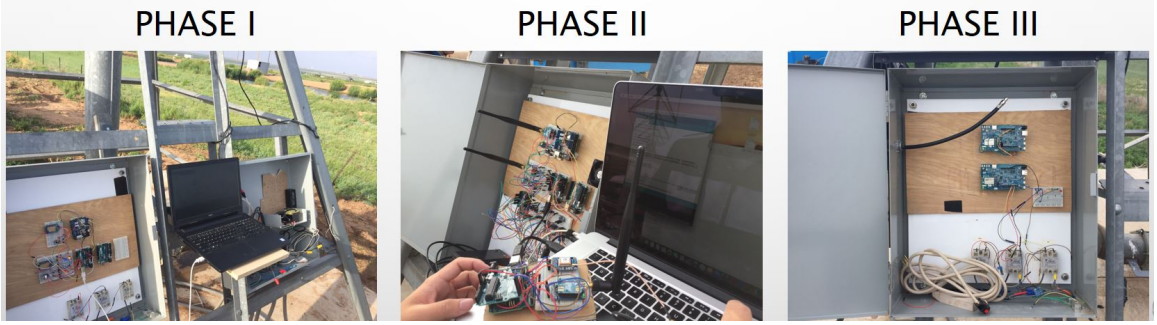


Figure 5.8: Control subsystem prototypes.

In real practice, users often need to move the pivot to some specific angle or location. The Test/Manual gives an user full control of the system. The user can tell the pivot to move at either forward or backward direction, adjust its moving speed by setting the engagement ratio from 0 to 100%(full speed). Users can also specify where he/she wants the machine to stop, and if or not to turn on water while moving.

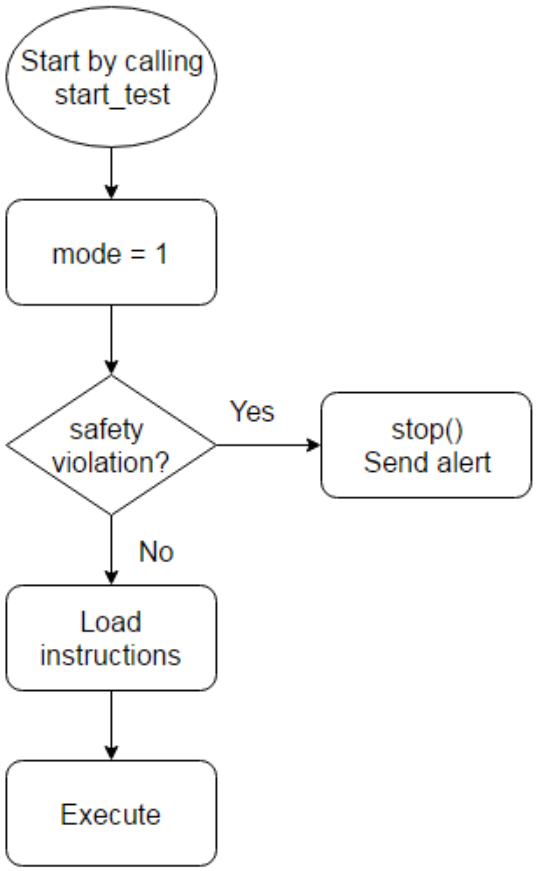


Figure 5.9: Flow of Test/Manual mode.

In figure 5.9When the controller receives the command to start a test/manual operation, it sets the mode to 1, and checks if there is any safety violation. For instance, the machine is outside of operational area, and the received direction tells the machine to go further beyond, then there is a potential risk. The controller will stop and send out an alert about the violation to the user. If no violation is found, controller will load the instructions, and execute accordingly.

In the User Programmable mode, users can customize an irrigation plan, store it and execute on a specific date. On the web app, users can divide the field to however many zones as they want, and specify how much water shall be applied to each zone. Up to 5 different plans can be stored. This allows for crop-specific management where multiple crops with different water needs are grown in the same field. Also, it is of great importance for scientific research facilities. Therefore, with

User Programmable mode, researchers from different groups can keep their irrigation "recipes" on the controller, and easily manage their own zones without interfering with other groups. Existing plans are displayed with their names and specifics. Users can update their settings if they want to.

In the Planned Automatic mode, the field is divided into 9 zones, each covering 30 degrees. By selecting the check box, users can activate editing irrigation operation on each zone. There are three submodes that users can choose from: Full Auto, Threshold, and Regular Periodic. In Full Auto mode, users don't need to do anything but sit back and relax. The controller downloads latest weather data and monitors soil sensor data constantly. Inside, it stores a look-up table that is generated by the Reinforcement Learning Algorithm that we discussed in section 4. By looking at the table, the controller knows when to activate the irrigation machine and how much water is needed for each zone in the field. The controller is also designed to run other common irrigation methods such as threshold and regular periodic. In Threshold mode, users can define up to 3 different MAD levels. The irrigation is triggered whenever the soil moisture level drops below the predefined MAD levels, and the zone is filled up to the field capacity. In Regular Periodic mode, each zone is irrigated periodically, for example, every 4, 5, or 7 days, with a fixed amount specified by the engagement ratio.

In both User Programmable and Planned Automatic mode, fields are divided by zones. We wish to treat each zone as an entity. Therefore, the speed of the pivot over a particular zone should be a fixed value. However, soil moisture values change over time, triggering speed change during an irrigation. In order to be consistent, the speed of the pivot over a particular zone is calculated and fixed at the moment when the pivot is just about to enter the zone.

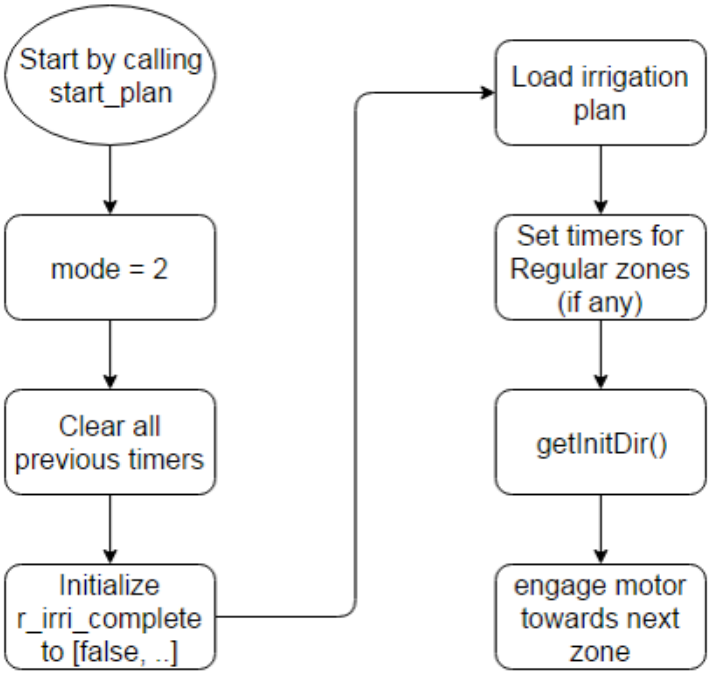


Figure 5.10: Initialization sequence of Planned Automatic mode.

Another common problem is related to the starting position. Since we must treat any zone uniformly, we cannot allow irrigation to start at any random location. Instead, any irrigation has to be started precisely at a zone boundary. A set of automatic initialization movement that aims to align the pivot with the closest zone boundary is designed. The first step is to find out which direction to go. As shown on Figure 5.11, the controller calculates relative distances between its current location and adjacent two zone boundaries. Then set the direction towards the closest one, engage the motor to move with full speed.

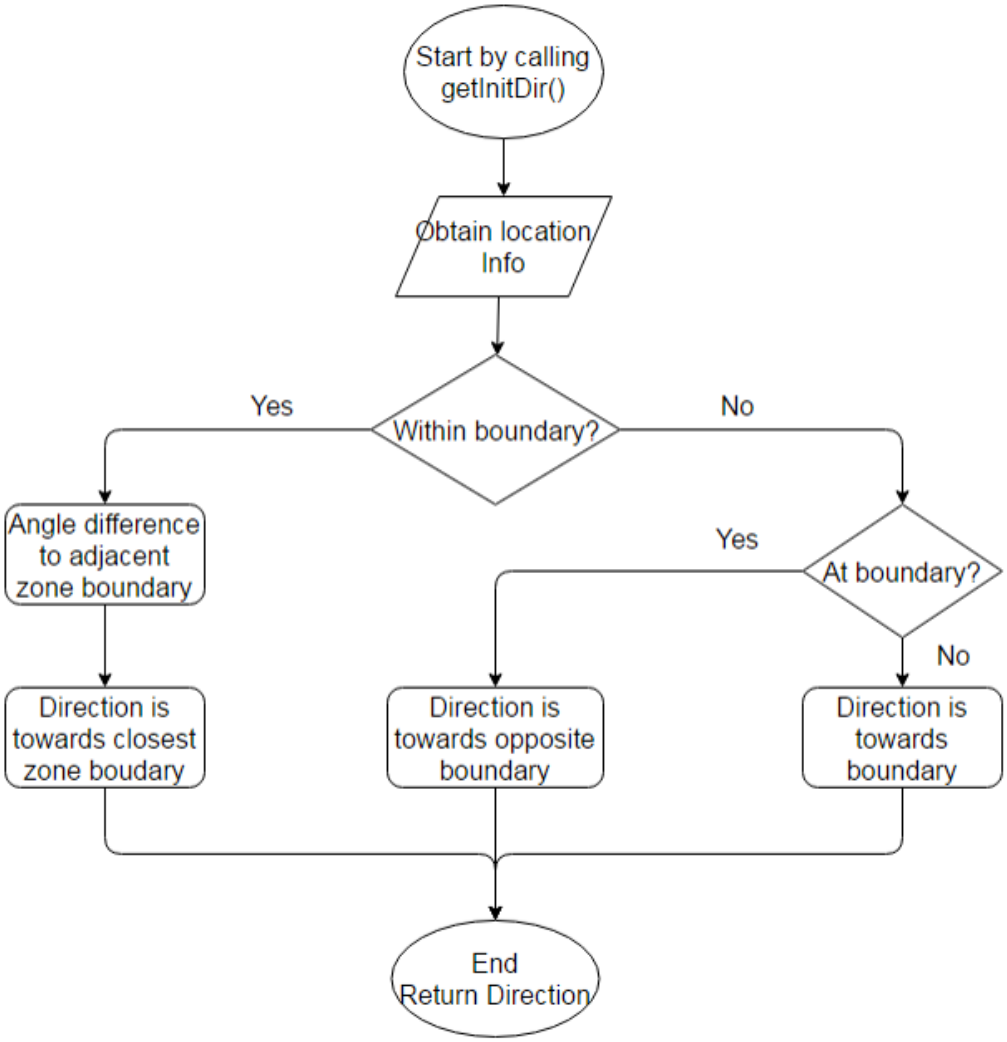


Figure 5.11: Initial direction is found by comparing relative distance to adjacent zone boundaries.

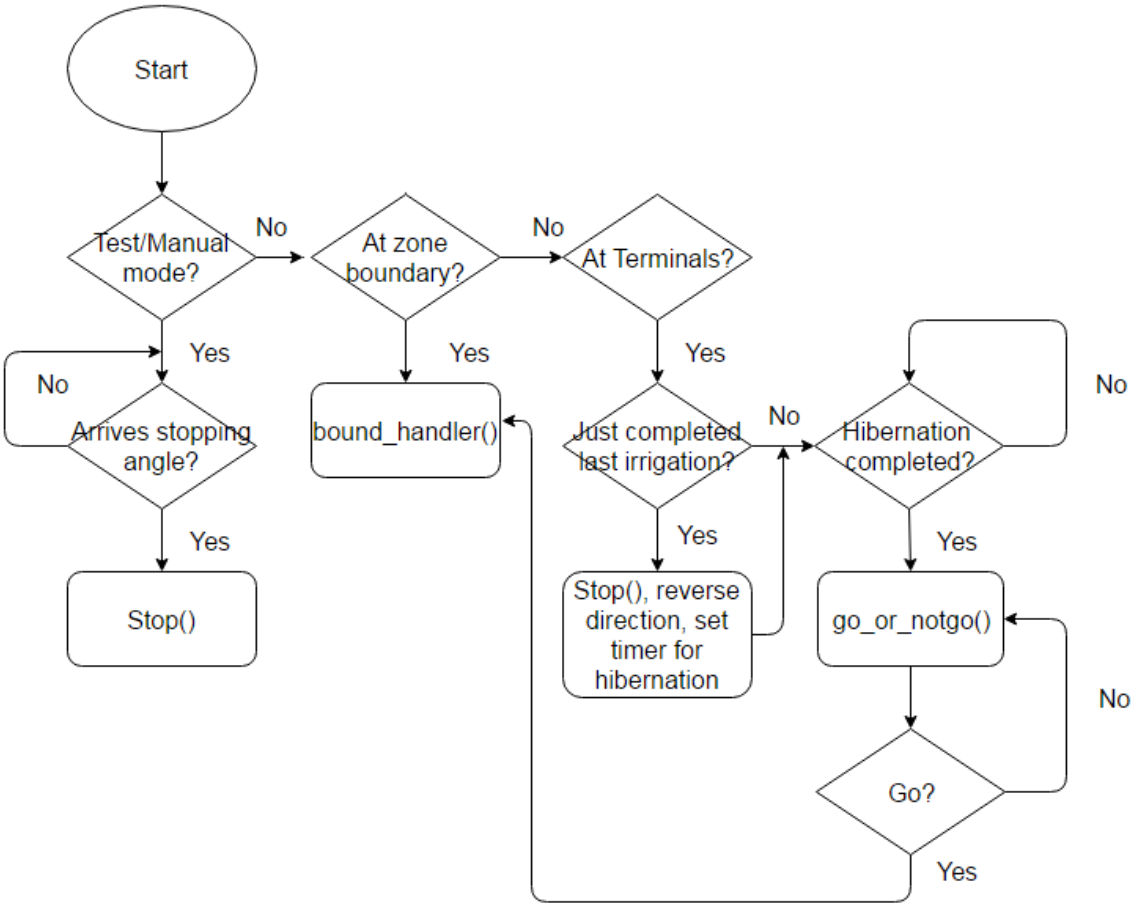


Figure 5.12: Controller constantly checks pivot locations against zone boundaries and terminal position.

Ideally, the field that a center pivot irrigation machine runs on is a full circle. However, sometimes, fields are irregular or just not big enough for center pivot to run a full circle. So, a pair of terminals are set (in degrees) to define the operational range of the center pivot, as shown on figure 5.12. For example, 0 to 180 degree. In both User Programmable and Planned Automatic mode, the controller constantly checks its current locations. If a zone boundary is reached, it loads the specification of the next zone, and changes the speed if necessary. When a terminal is reached, the machine is put into "hibernation", under which the machine stops, turns off water, and waits for the reactivation, as shown on figure 5.13 and 5.14.

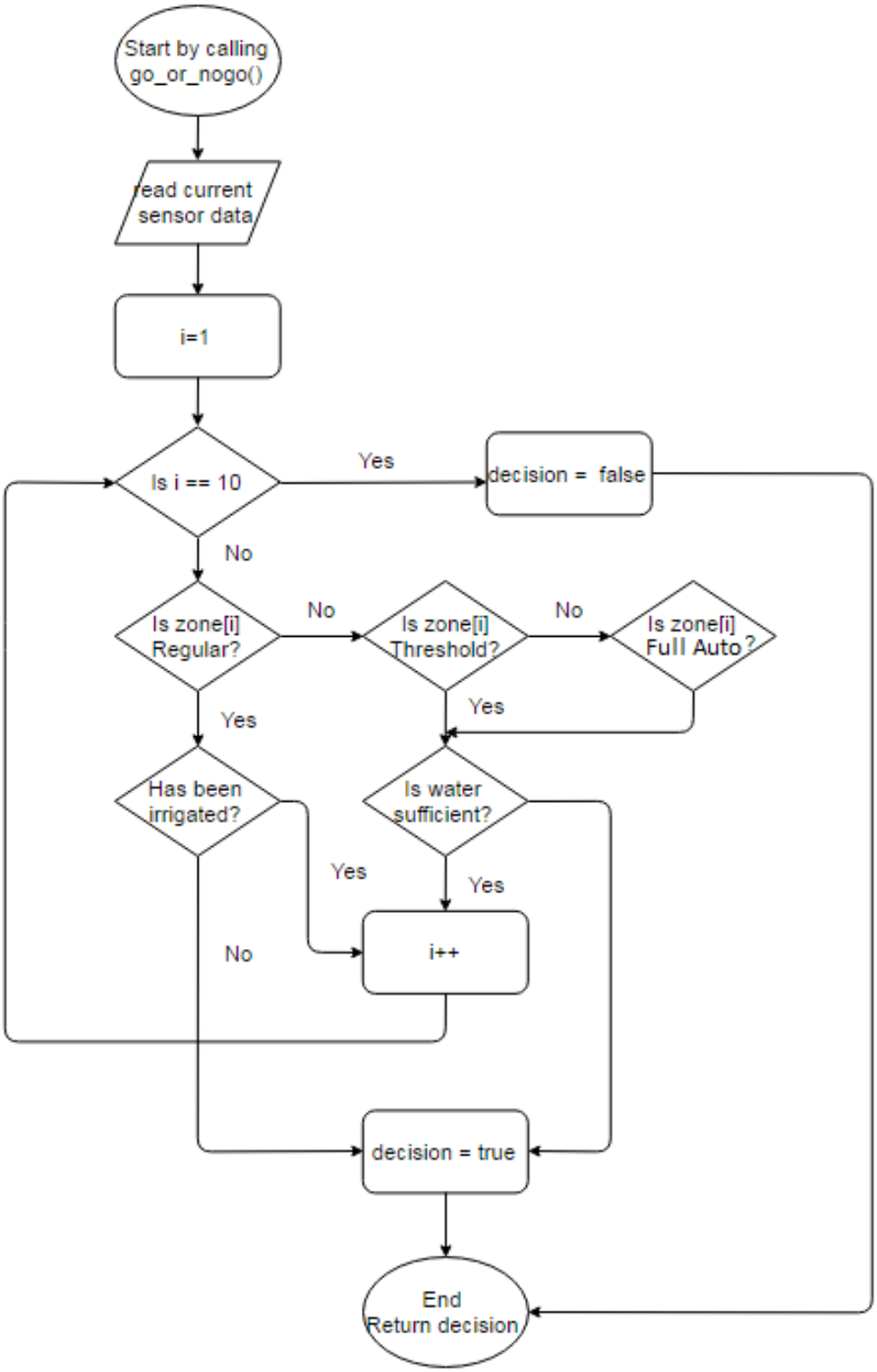


Figure 5.13: Whether a new round of irrigation is activated nor not, under Planned Automatic mode, is decided by scanning all zones for soil moisture level change. In Full Auto mode, the decision making also involves weather information.

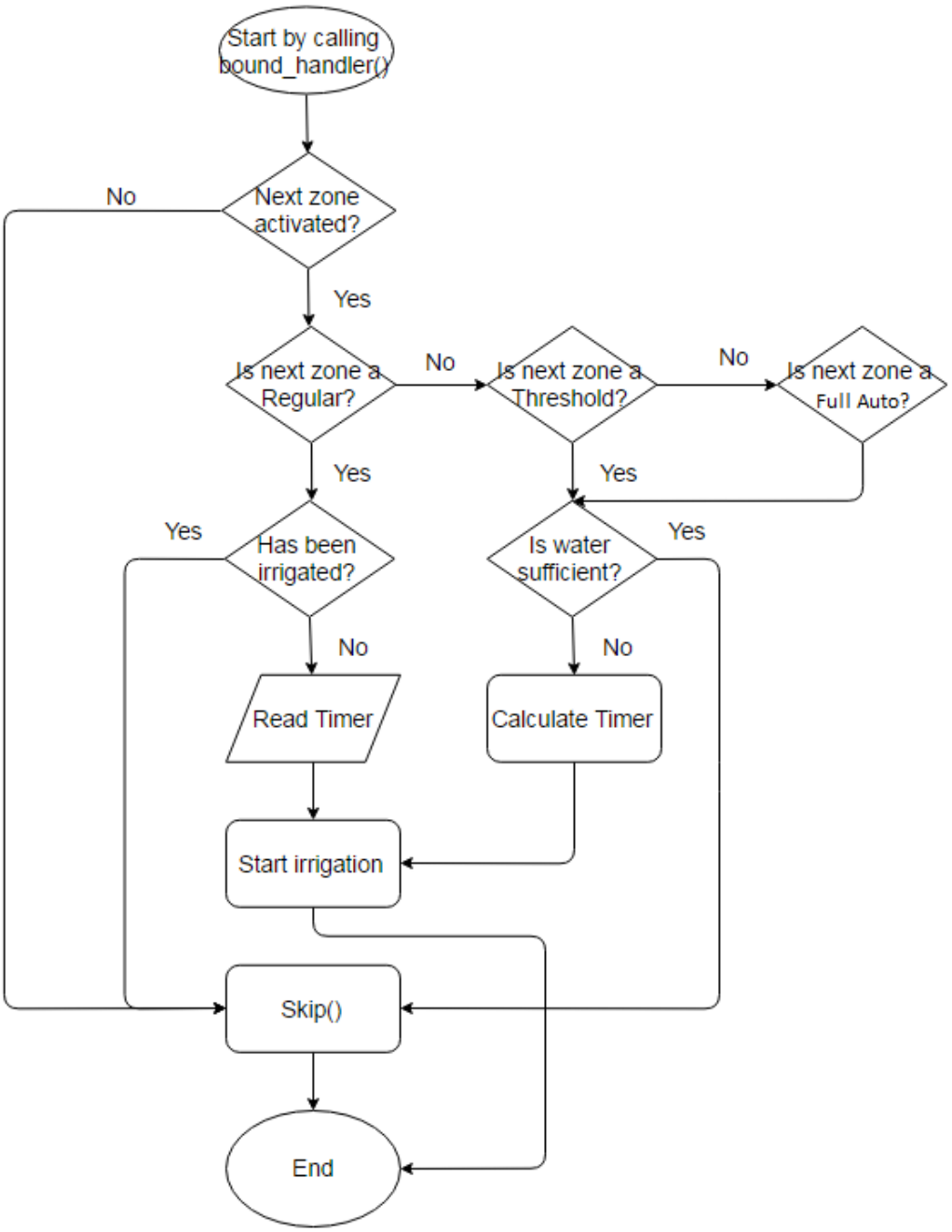


Figure 5.14: Each time the pivot reaches a zone boundary, the controller loads the specifications of the next zone, and adjusts speed and water on/off if necessary.

5.3 Pivot Movement

A center pivot irrigation machine releases water while it is sweeping over a field. The total amount of water applied is proportional to the time it spends over the field. For example, our job is irrigate a field of 360 degree circle. If the flow rate is 0.04 m^3/s, and it takes an hour to irrigate an area of 14000 m^2. Then the total amount of water is 0.04m^3/s $\times$ 3600s = 144 m^3, which is equivalent to 144$m^3 \div$ 14000m^2 = 0.01m = 1cm in depth.

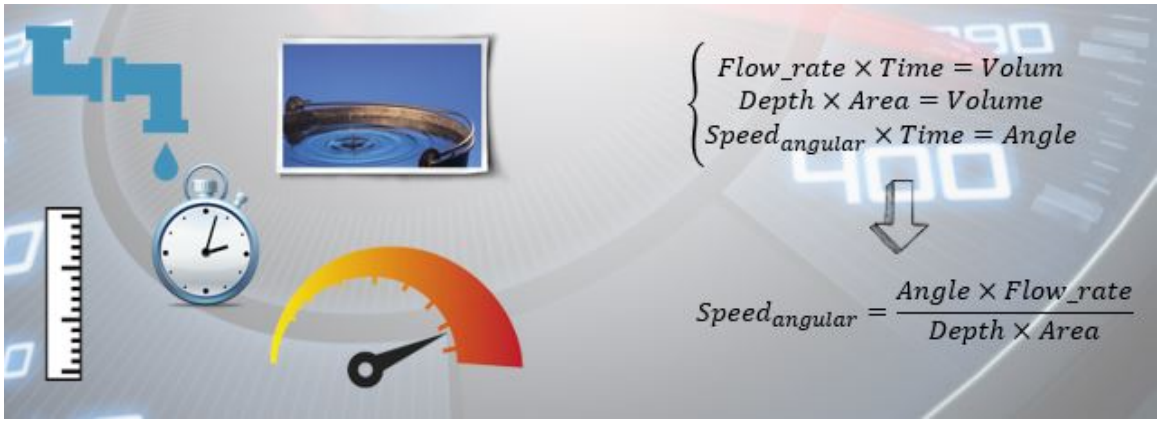


Figure 5.15: How pivot speed is calculated.

Let's say we know that the full speed of a center pivot machine is 0.1 degree/s, and the flow rate is 0.04 m^3/s. If our goal is to apply 2.0 cm of water to the same field, how can do we set the speed? Simple! As the figure 5.15 suggests, the amount of water pumped through the pipe is equivalent to the irrigation depth multiply by the irrigated area. 2.0cm depth of water to a 14000 m^2 field is equivalent to 0.020m $\times$ 14000m^2 = 280 m^3, which takes 200 $m^3 \div$ 0.04 m^3/s = 7000s = 1.94hours to fill up. Since a full circle is 360 degree, the speed should be 360 degree $\div$ 1.94 hours = 0.05 degree/s. Therefore, we can reduce our speed to 50% of the full speed. In reality, the pivot can only move with a constant speed. The way to slow down is to turn off the engagement half of the time. For example, in this case, 50% time off means that in every minute, the pivot is only going to move 30 seconds, and it stops for the other 30 seconds, then it will move again

for 30 seconds, and rest for another 30 seconds, so on so forth. The engagement of on and off is controlled by a contactor (figure 5.16), and the percentage is defined as the "engagement ratio". Note, there is a minimum depth of water a machine can apply since it cannot go any faster. A faster way to calculate the engagement ratio is to divide minimum depth by require depth.

Figure 5.16: A 40A solid state relay used in the control box.

Figure 5.17 demonstrates hardware connection of all our speed control components. A center pivot machine is powered by electrical motors, which use 3 phase 480v AC. We can change the direction by simply reversing two of its three stator leads. Alternatively, we can mount 2 sets of leads, each for one direction, forward and reverse. At any given time, only one lead is connected to the motor. Each lead goes through a contactor, which is an electrical-mechanical switch. The contractors are run by 120V AC, therefore cannot be directly controlled by any MCU whose typical output voltage is 3.3-5 V DC. Thus, we bring in solid state relays (SSD) to fill in the gap. A SSD is an electronic switch that can be controlled by small voltage. Another great feature of a SSD is that it can tolerate large current which occurs during the on/off transitions.

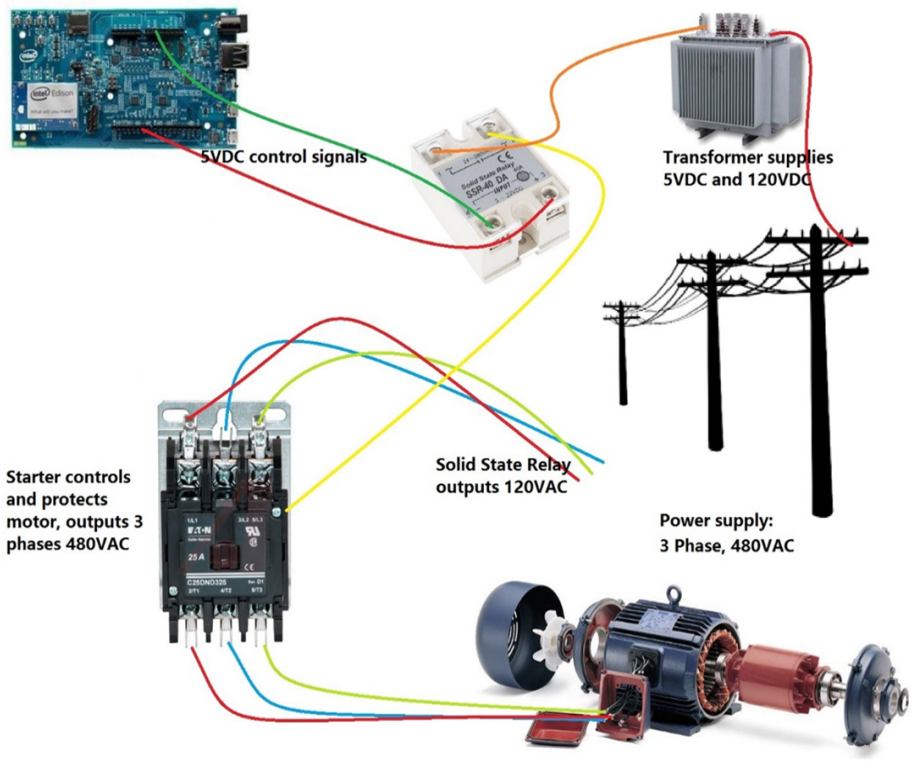


Figure 5.17: How the speed and directions are controlled.[11]

In reality, the precision of pivot speed might be less than ideal due to reasons such as slope or mechanic slippage, rough terrain, or mechanical failures. Therefore, an adjustment algorithm is needed to make certain corrections. A proportional integral derivative (PID) mechanism was added to the control algorithm (figure 5.18). It calculates an error term $e(t)$ as the difference between actual observed speed and the ideal speed. Then, as the name suggests, the proportional, integral, and derivative terms of the error w.r.t time are combined as the new output.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt' + K_d \frac{de(t)}{dt} \tag{5.1}$$

Where K_p, K_i, and K_d are non-negative coefficients for the corresponding terms. In the control algorithm, K_p, K_i, and K_d are all set to be equal to 1. To avoid unnecessary complication, the PID is only activated when the error is larger than 5%. This feature allows significant improvement

over currently available commercial pivot controllers.

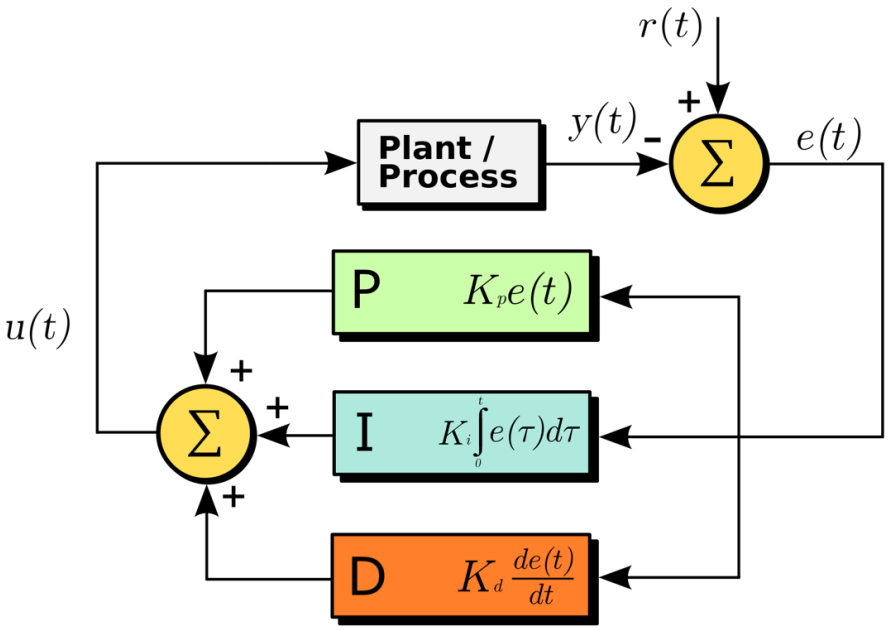


Figure 5.18: A block diagram of a PID controller in a feedback loop [12].

5.4 User Interface

As introduced before, users can choose to control the system on site by a manual interface or through a web app remotely. Figure 5.19 and 5.20 show the circuit board we used for manual control in early and later phases respectively. The manual control interface is designed to fulfill basic operations. LED lights are used to display operation status. Buttons and a potentiometer were added to provide discrete and continuous input signals. Extra pins were added to provide extendability so that new functions can be easily added to the manual interface later. The web app, however, was designed to be much more powerful. In the app, users can run the irrigation system in three modes: Test/Manual, Planned Automatic, and User Programmable. The controller uploads its current operational data every 2 seconds.

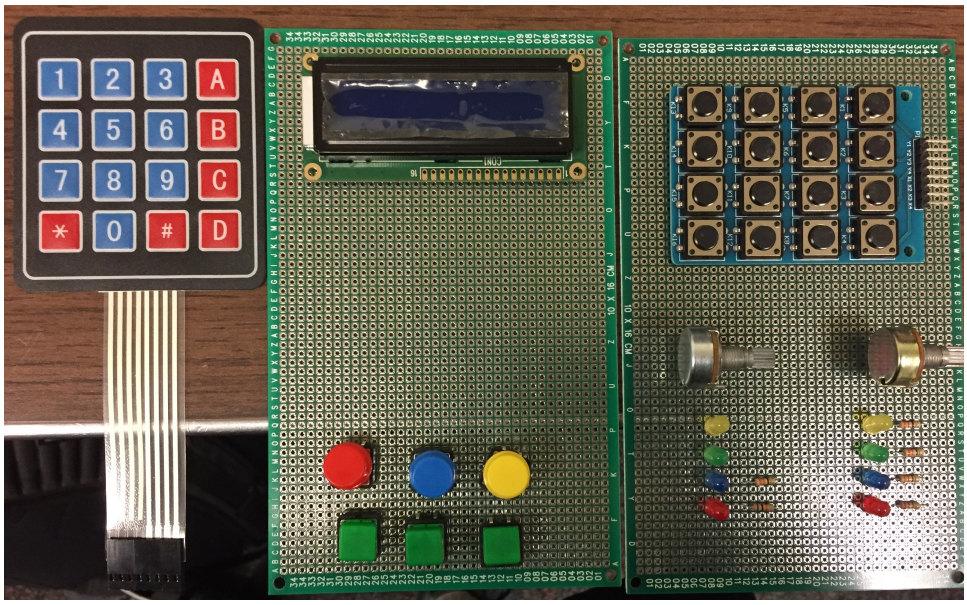


Figure 5.19: Early manual control interface. There were later replaced by PCB boards.

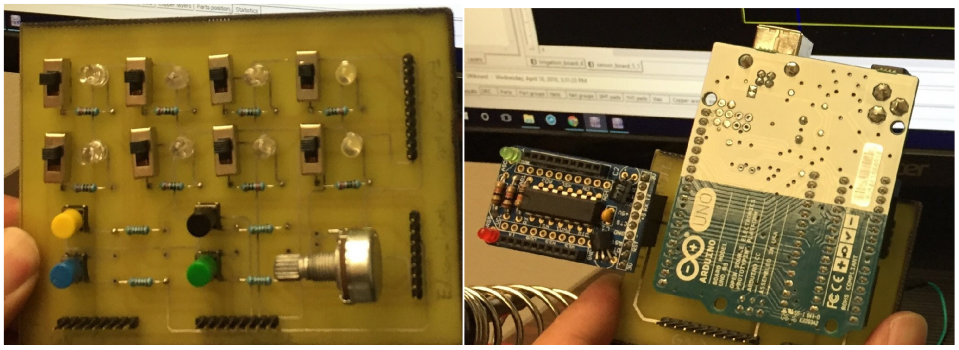


Figure 5.20: PCB boards. The left one is for manual control at center pivot, the right one is for sensor box.

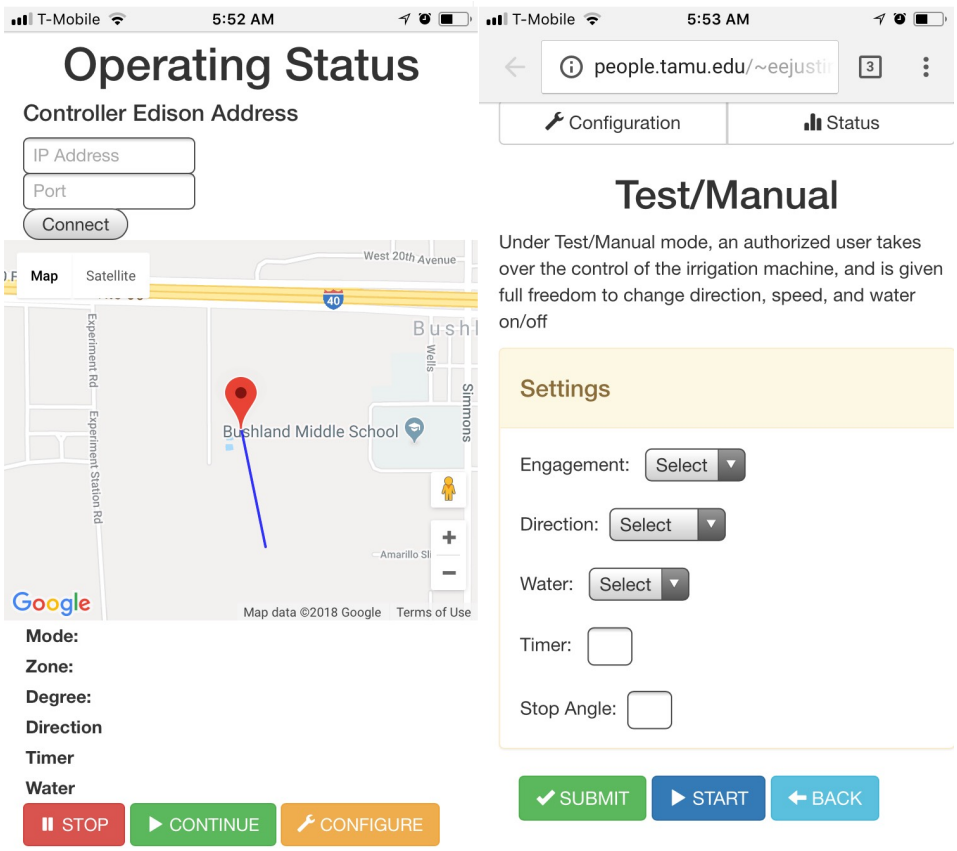


Figure 5.21: User interface at Operating Status

Shown on Figure 5.21, an authorized user can receive real-time operational status on a smart phone, a tablet, or a laptop computer. A Google map displays real-time position of the center pivot machine. A user can stop the machine whenever he/she deems necessary. He/she can resume previous operation by pressing continue button. The "Configure" button navigates the user to the configuration page if any changes need to be made.

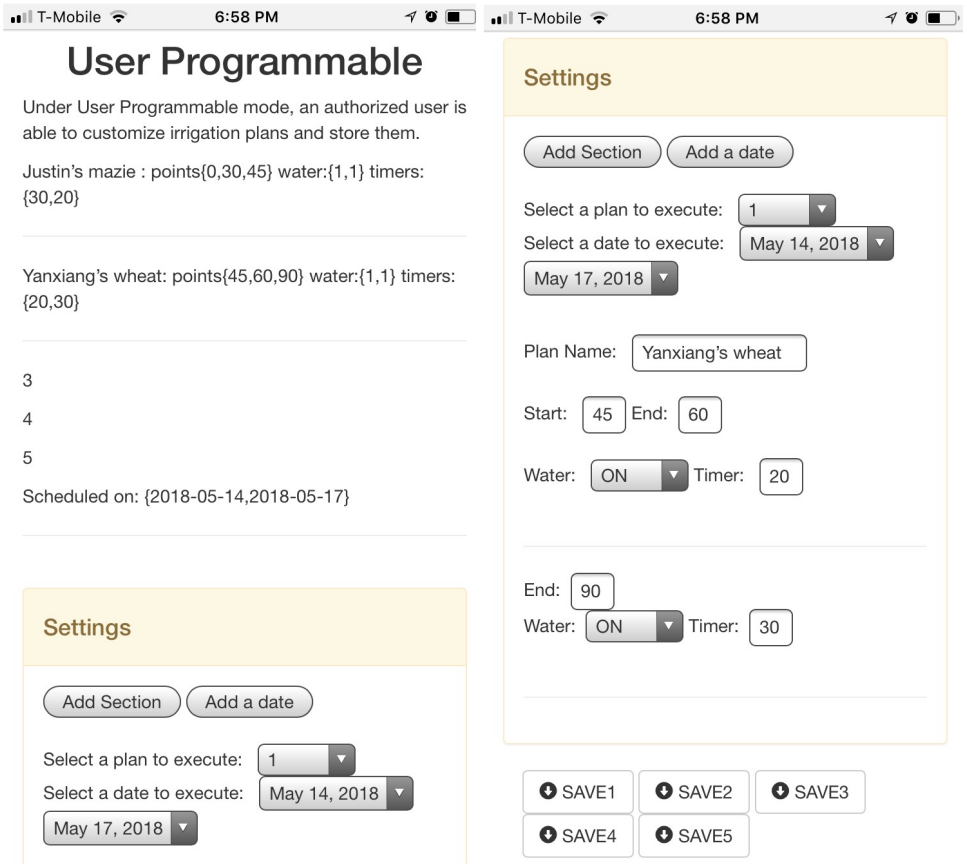


Figure 5.22: User interface at Programmable mode

In the User Programmable page, users can find existing plans and their settings. For example, on Figure 5.22, there are two plans: Justin's maize and Yanxiang's wheat. We can also find that, in Justin's plan, the field is divided into 0-30 degree, and 30-45 degree. The plan is to irrigate 0-30 degree with the speed of 30% of engagement ratio and 30-45 degree with 20%. The plan is scheduled to be executed on May 14, 2018 and May 17, 2018. On the other hand, Yanxiang's field is from 45 degrees to 90 degrees, also been divided into two sections. They are to be irrigated with a different speed. Users can store up to five plans.

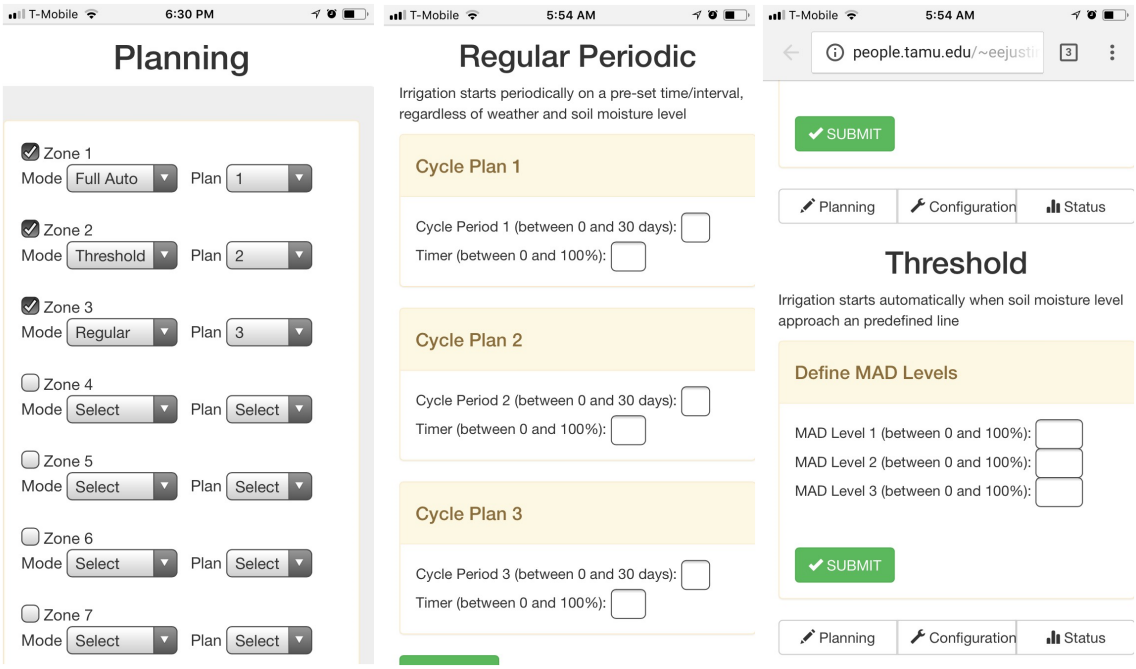


Figure 5.23: User interface at Planned Automatic mode

In Planned Automatic mode, shown on figure 5.23, users have the freedom to activate some or all 9 zones for irrigation. Unselected zones will be skipped during an irrigation. For each zone, users can choose one of three submodes. In addition, in Threshold mode, users can also define up to three threshold values. In Regular Periodic mode, we are able to adjust the length of an irrigation period, and specify the amount of water to be applied. Figure 5.24 shows the moment when the author was demonstrating the user's app to control the operation in the field.

Figure 5.24: The author was demonstrating how to use the web app on a smart phone.

5.5 Security

Several layers of security were added to protect the system and the data from cyber-attacks.

1. Any user must login with password.
2. To connect to the controller, an user must know the correct port number, acting as the second "password".
3. Operational data are not stored on the "control center", but kept separately on the "data center", which is not directly accessible.
4. To prevent hacking from outside, the web-app is hosted on tamu.edu domain, therefore, under the university cyber security umbrella.
5. In addition, the controller is connected to the "AgNet" (agnet.tamu.edu), which distributes local IP addresses to the Edison boards. Thus, controllers are effectively shielded by the "AgNet". Any user must possess a valid registered AgriLife account in order to connect to

the controller. The I.T department at Bushland clears inactive AgriLife accounts every 90 days. This effectively reduces the risk that an expired account being used by any attackers.

6. MODEL FUSION

6.1 Introduction

Crop yield simulation models, like the well-known DSSAT, AquaCrop, APSIM, and WOFOST, are widely used in today's agricultural research. These models carry deep understandings of crops growth that were derived from decades of research and experiments on soil water, temperature, and plant response to environmental conditions, etc. Many of them also take management factors into consideration, including irrigation, fertilization, and tillage.

Although these models represent a state of the art of knowledge in plant biology [73], their abilities to produce accurate yield predictions are, in many cases, less than ideal. Researchers often need to calibrate their models with the observation data [74]. These calibration processes are not only slow but also inefficient. Especially when dozens, or even over a hundred, of relevant parameters need to be tuned, the work load is immense. What makes the matters worse, is that every calibration works only for a specific site and season. Therefore, it is extremely difficult and unreliable to use those models at large scale.

In section 3, we discussed weather forecast. In meteorology, ensemble forecast is a common practice, in which researchers collect predictions from different predictive models, and combine these outcomes to generate a new forecast. Under the similar thought process, our idea is to utilize the collective wisdom from multiple crop yield models, and build a convenient framework that not only able to produce accurate and reliable yield predictions, but also fast and easy to use at large scale without worrying about the time-consuming calibration process.

In fact, there is an initial attempt in such direction done by James Rising and Mark Cane [15]. In an unpublished work, data from 15 global soil, precipitation, elevation, fertilization, irrigation, and harvest databases are used to simulated crop yields on six crops around the world, using DSSAT and AquaCrop models. Simulation results are compared with actual yield data recorded by FAO and the USDA Foreign Agricultural Service. They found that AquaCrop performed relatively

Table 6.1: Correlations for each combination of crop and model [15]

Crop	AquaCrop	DSSAT	Average	Fitted	Corrected
Barley	0.10	0.20	0.35	0.38	0.93
Maize	0.24	-0.19	0.22	0.25	0.90
Rice	-0.27	0.00	-0.13	0.26	0.85
Sorghum	0.14	-0.24	-0.02	0.36	0.86
Wheat	-0.15	0.05	-0.15	0.15	0.92

well in northern latitudes but exhibited low or even negative correlation in Brazil and sub-Saharan Africa. Its performance also varies by crop types. On the other hand, DSSAT can capture Africa and temperate regions very well. Yet, its predictions are not good for certain crop like wheat. Recognizing the raw predictions from these two models are poor, they consequently built three statistical combinations of DSSAT and AquaCrop predictions: "Average", "Fitted", and "Corrected". The "Average" is simply taking the mean of two estimates on a country-wide basis. The "Fitted" method used a linear model:

$$y_{it} = \beta_{AquaCrop} y_{it}^{AquaCrop} + \beta_{DSSAT} y_{it}^{DSSAT} + \gamma_t + \epsilon_{it} \tag{6.1}$$

where y_{it} is the yield for country i in year t, and $\beta_{AquaCrop} y_{it}, \beta_{DSSAT}$ are the coefficients assigned to each model. γ_t is a constant term that is yearly specific. ϵ_{it} is the error term for country i in year t. Lastly, the "Corrected" method is a quadratic function of the corresponding prediction of the two models' outcomes. The performance, measured by data correlations between prediction and actual yield is summarized in the Table 6.1.

From the above findings, it's safe to say that models can complement each other. Where AquaCrop performed poorly, it can be supplemented by DSSAT, and vice-versa. But, how? Recall that, in section 5, a neural network is constructed to replace the bulky DSSAT for faster simulation on yield. The network used daily total soil water level during the crop season as inputs, and showed very inspiring high accuracy [75]. It is a living proof that machine learning techniques could have great potential in crop yield predictions.

Table 6.2: The datasets used as inputs for AquaCrop and DSSAT

Data Type	**Data Source**
Soil Texture	HWSD
Weather	NNDC Climate
Geographic	GLOBE
Harvest	FAOSTAT

6.2 Preparation of Data

Both DSSAT and AquaCrop need detailed input data for simulations. We used the following global datasets for soil, weather, geographic, and harvest data, as summarized in Table 6.2.

Four of world's most produced agricultural crops are simulated, and according to FAOSTAT, top five countries in each crop, during the perriod from 1986 to 2015, were selected. In each country, the total harvest area (in hectares) for a specific crop in each year were add up from 1986 to 2016. An example is shown in Table 6.3. A manageable number of stations are decided approximately in proportional to each country's accumulative harvest area for each simulated crop, as shown on figure 6.1. The locations of experimental fields are picked under three basic considerations:

1. Weather and soil data should be available, and as complete as possible at each location.
2. The distribution of stations for each country should be as wide and even as possible.
3. Locations should be consistent with crop production reality.

Following the above rules, no stations are selected in the state of Alaska, United States, because, according to the agricultural production reality, Alaska is too cold for most of the crops to grow. Similarly, only a few stations are selected in northwest part of China since this area is mostly desert. On the other hand, most stations selected for Brazil are along the coast. It is not because of the agricultural reality, but simply due to poor quality (months or even years of missing data) of weather data in Brazil's inland stations. The overall poor quality of weather data are also a serious problem in India. However, luckily, it is largely compensated by sufficient number of stations available across the country. Therefore, the overall distribution of stations in India is wide

and even.

Figure 6.1: Simulation locations

Table 6.5 demonstrates the information contained in a raw data sheet downloaded from NNDC Climate database [76]. STN is the global station identification number. YMD marks the date of the record, TEMP is the mean temperature for the day in Fahrenheit degrees, MAX and MIN are the maximum and minimum temperature observed for that day, * indicates that the MIN is derived from the hourly data. DEWP is the mean dew point temperature. SLP tells the mean sea level pressure for the day. VISIB is the mean visibility in miles. WDSP and MXSPD are the mean and maximum wind speed. PRCP is the total precipitation (rain or melted snow) in inches, 99.9 means missing, D indicates the data are calculated from summation of 4 reports of 6-hour precipitation amount.

Table 6.3: Maize harvest area (in hectares) in U.S, China, Brazil, Argentina, and India, from 1986 to 2015 [16].

Year	US	China	Brazil	Argentina	India
1986	27885008	19198886	12460130	3231000	5923100
1987	24080000	20290803	13499440	2900000	5560900
1988	23573008	19773891	13181990	2437500	5896700
1989	26216000	20433871	12918980	1683700	5915100
1990	27094800	21482665	11394300	1560330	5904300
1991	27851580	21648530	13063700	1900100	5859400
1992	29168840	21119959	13363609	2365440	5962500
1993	25468360	20770936	11869663	2503010	5995000
1994	29345690	21229239	13748813	2445040	6135800
1995	26389830	22848467	13946320	2521750	5979000
1996	29398300	24571227	11975811	2603720	6300000
1997	29409230	23836937	12562130	3410385	6321000
1998	29376040	25281404	10585498	3185390	6203700
1999	28525380	25938564	11611483	2514650	6422100
2000	29315740	23086228	11890376	3088715	6611300
2001	27829720	24310506	12335175	2815504	6581500
2002	28057160	24660837	11760965	2420124	6635200
2003	28710330	24092820	12965678	2322857	7343400
2004	29797730	25467145	12410677	2338602	7430400
2005	30399100	26379450	11549425	2783436	7588300
2006	28586490	28482649	12613094	2447166	7894000
2007	35013780	29496901	13767400	2838072	8117300
2008	31796490	29882708	14444582	3412155	8173800
2009	32168810	31203367	13654715	2353175	8261600
2010	32960380	32517988	12678875	2904035	8553200
2011	33944990	33559864	13218892	3747838	8780000
2012	35355740	35046465	14198496	3696300	8710000
2013	35390550	36339411	15279652	4863801	9430000
2014	33644310	37150397	15432909	4836655	9258000
2015	32678310	38147048	15406010	4626880	8690000

Table 6.4: Harvest area for each simulated crop and the number of stations selected for simulations

Maize					
	US	China	Brazil	Argentina	India
Harvest Area	29823798.26	26684796.48	13056375.81	2971094.29	7181825.806
# stations	30	27	13	30	7
Wheat					
	China	India	U.S.	Russia	France
Harvest Area	26515826.1	26515958.97	21858734.77	23303722.76	5121176.839
# stations	27	27	22	23	5
Soybean					
	U.S.	Brazil	Argentina	China	India
Harvest Area	27875368.58	17482978.32	11016737.32	8276784.774	6708432.903
# stations	28	17	11	8	7
Tomato					
	China	India	U.S	Turkey	Egypt
Harvest Area	689364.4516	501613.9032	170310.4194	182952.6129	188069.1935
# stations	7	5	2	2	2

Table 6.5: Part of raw weather data from station 584570 at Huangzhou China.

STN—	YMD	TEMP	DEWP	SLP	VISIB	WDSP	MXSPD	MAX	MIN	PRCP
584570	19860101	37.1	26.1	1028.8	6.1	2.4	5.8	43.9	30.9*	0.00D
584570	19860102	38.8	27.7	1027.6	5.1	3.9	5.8	45.5	28	0.00D
584570	19860103	41.9	28.4	1026.2	6	3.6	9.7	46.8	34.7	0.00D
584570	19860104	30.5	13.4	1033.3	7.7	8	11.7	41	22.5*	0.00D
584570	19860105	28.2	8.3	1031.9	10.4	3.9	7.8	34.0*	20.5	0.00D
584570	19860106	35.8	15	1025.6	7	4.6	5.8	43.2	23.5	0.00D
584570	19860107	48.9	22	1025.5	8.6	9.6	14	57.2*	33.3	0.00D
584570	19860108	36.5	13.9	1032.2	9.2	2.4	5.8	45.3	29.3*	0.00D
584570	19860109	39.4	23.9	1031.4	4.7	2.9	5.8	51.8	27.7	0.00D
584570	19860110	37.8	29.5	1032.7	3.8	4.1	5.8	47.1	30.7*	0.00D

One of the challenges in data collection is the ET data. As you may have already discovered, the ET data are not available in weather database. Rising and Cane[15] used a monthly climatology ET data from FAO GeoNetwork. The problems are: first, monthly data are far from accurate. The simulations of AquaCrop needs daily ET data, while although DSSAT does not require ET as an input, the model itself calculates ET as part of the essential part in the simulation. Therefore, in our work, the daily ET values are generated according to FAO Penman-Monteith Equation [13]:

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma\frac{900}{T+273}u_2(e_s - e_a)}{\Delta + \gamma(1 + 0.34)u_2} \tag{6.2}$$

where ET_o is the reference evapotranspiration [$mm\ day^{-1}$], R_n is net radiation at the crop surface [$MJ\ m^{-2}day^{-1}$], G is soil heat flux density [$MJ\ m^{-2}day^{-1}$], T is air temperature at 2 m height [$^{\circ}C$], u_2 is wind speed at 2 m height [$m\ s^{-1}$], e_s is saturation vapour pressure [kPa], ea is actual vapour pressure [kPa], $e_s - ea$ is saturation vapour pressure deficit [kPa], Δ is slope vapour pressure curve [kPa $^{\circ}C^{-1}$], γ is psychrometric constant [kPa $^{\circ}C^{-1}$].

By the first look, none of the above parameters are directly available from our raw weather data. But fortunately, they can be derived.

Firstly, the psychometric constant γ, is given by:

$$\gamma = \frac{c_p P}{\varepsilon\lambda} = 0.665 \times 10^{-3} P \tag{6.3}$$

where P is atmospheric pressure [kPa], λ is the latent heat vaporization, 2.45 [MJ Kg^{-1}], c_p is the specific heat at constant pressure, 1.013×10^{-3} [MJ $Kg^{-1}\ ^{\circ}C^{-1}$], ε is the ratio molecular weight of water vapour in dry air = 0.622.

Next, saturation vapour pressure e_s is a function of temperature T:

$$e(T) = 0.6108exp(\frac{17.27T}{T + 237.3}) \tag{6.4}$$

The mean saturation vapour pressure is the mean of the value taken at maximum and minimum temperatures of a day:

$$e_s = \frac{e(T_{max}) + e(T_{min})}{2} \tag{6.5}$$

The actual vapour pressure e_a is derived from dewpoint temperature:

$$e_a = e(T_{dew}) = 0.6108exp(\frac{17.27T_{dew}}{T_{dew} + 237.3}) \tag{6.6}$$

Since the wind speed is normally measured at 10 meters' height, we need to convert the speed to readings at 2 meters in order to fit in the FAO Penman-Monteith Equation. The conversion can be done by equation 7.7:

$$u_2 = u_z \frac{4.87}{ln(67.8z - 5.42)} \tag{6.7}$$

where u_2 and $u_z$is the wind speed at 2 and z meters above ground respectively.

The slope of saturation vapour pressure curve Δ is also a function of air temperature, given by:

$$\Delta = \frac{4096[0.6108exp(\frac{17.27T}{T+237.3})]}{(T + 237.3)^2} \tag{6.8}$$

The calculation of net solar radiation (R_n) is complicated since the length of the daytime of any given location changes over a year. Therefore, the solar radiation is not only a function of a location's latitude, but also a function of time.

Let's start from the very basic. Conceptually, R_n is the difference between net shortwave radiation (R_{ns}) and reflected net long-wave radiation (R_{nl}):

$$R_n = R_{ns} - R_{nl} \tag{6.9}$$

where R_{ns} is a fraction of incoming solar radiation (R_s):

$$R_{ns} = (1 - \alpha)R_s \tag{6.10}$$

The coefficient α is the canopy reflection coefficient. In our calculation, for simplicity, we take 0.23 across all simulations.

R_s is also subject to climate conditions, and can be adjusted by maximum and minimum temperature of a day.

$$R_s = k_{Rs}\sqrt{(T_{max} - T_{min})}R_a \tag{6.11}$$

where R_a is the extraterrestrial radiation [$MJ\ m^{-2}day^{-1}$], k_{Rs} is adjustment coefficient. For simplicity, all k_{Rs} in this paper are set to be 0.16.

The extraterrestrial radiation R_a is a function of latitude and time of a year, shown on figure 6.2, and calculated by:

$$R_a = \frac{24(60)}{\pi}G_{sc}d_r(\omega_s sin(\varphi)sin(\delta) + cos(\varphi)cos(\delta)sin(\omega_s)) \tag{6.12}$$

where G_{sc} is solar constant, which is 0.082 $MJ\ m^{-2}min^{-1}$,d_r is inverse relative distance between Earth and Sun, ω_s is sunset hour angle, φ is a location's latitude, δ is solar declination.

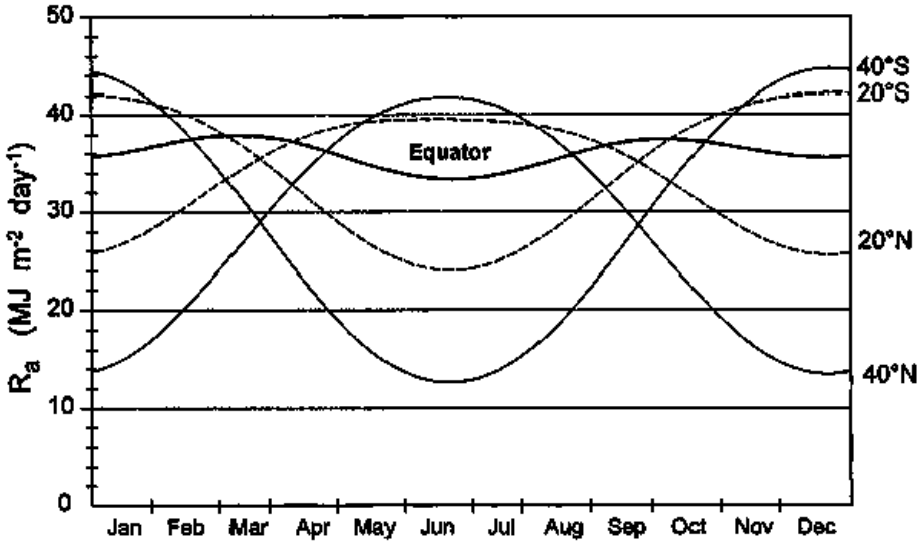


Figure 6.2: Annual variation in extraterrestrial radiation R_a at the equator, 20 and 40 degree north and south [13].

The inverse relative distance d_r, solar declination δ, and sunset hour angle ω_s are given by:

$$d_r = 1 + 0.033cos(\frac{2\pi}{365}J) \tag{6.13}$$

$$\delta = 0.409sin(\frac{2\pi}{365}J - 1.39) \tag{6.14}$$

$$\omega_s = arccos(-tan(\varphi)tan(\delta)) \tag{6.15}$$

where J is the number of the day in a year. For example, for Jan. 5, $J = 5$.

Finally, according to the Stefan-Boltzmann law [77], the long-wave radiation emission is a function of air temperature. Yet, due to substance like water vapour, clouds, and dust, some of the emission is absorbed. These factors can be quantified by humidity and cloudiness. Therefore, corrected net longwave radiation (R_{nl}) is given by:

$$R_{nl} = \sigma(\frac{T^4_{max,K} + T^4_{min,K}}{2})(0.34 - 0.14\sqrt{e_a})(1.35\frac{R_s}{R_{so}} - 0.35) \tag{6.16}$$

where, σ is the Stefan-Boltzmann constant, $T_{max,K}$ and $T_{min,K}$ are absolute temperature of a day, R_{so} is the clear-sky radiation, which is calculated by:

$$R_{so} = (0.75 + 2 \times 10^{-5}z)R_a. \tag{6.17}$$

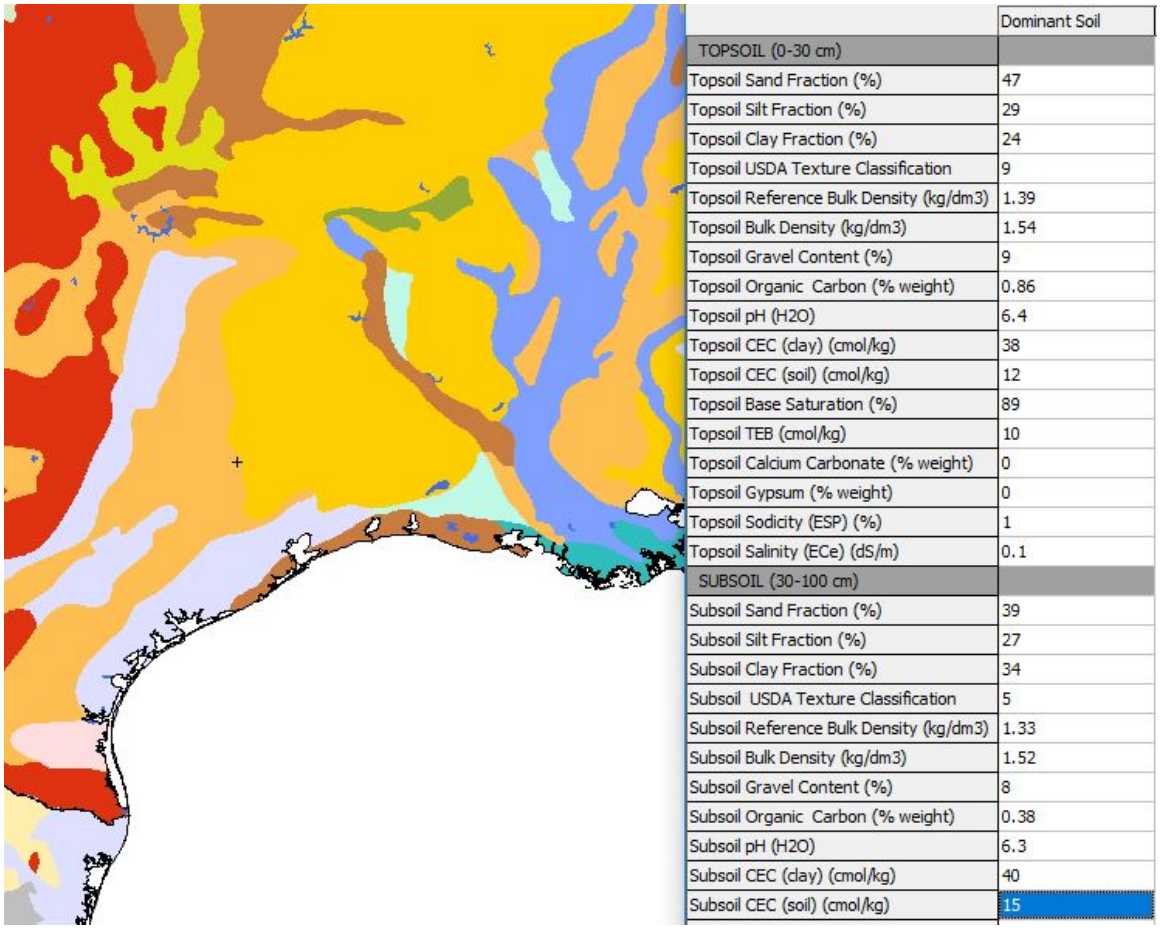
In which, z is the location's elevation in meters above sea level.

Using the equations above, daily ET values are calculated for all stations from 1986 to 2015.

Not all weather stations have the complete record for the span of 30 years. Missing data are common and inevitable when dealing with large quantity of historical data. Therefore, it is expected to miss several parameters in a day's record, or even have small number of days' of records missing. The rules in this project are the following:

1. Sporadic missing temperature data "-999" are filled with previous day's data if available.
2. Small number of Missing Precipitation data are filled with "0".
3. Entire missing days is recovered by fitting data generated by monthly average.
4. If over 3 months of data are missing, the station is abandoned. Replacement station nearby is selected, if there were any.

Soil data are obtained from the Harmonized World Soil Database (HWSD), which provided soil's composition at depth of 30cm and 100m, shown on figure 6.3. Some of the parameters required for running AquaCrop and DSSAT, such as Filed Capacity (FC), Permanent Wilting Point (PWP), and water content at saturation, can be derived from these the information from HWSD.



	Dominant Soil
TOPSOIL (0-30 cm)	
Topsoil Sand Fraction (%)	47
Topsoil Silt Fraction (%)	29
Topsoil Clay Fraction (%)	24
Topsoil USDA Texture Classification	9
Topsoil Reference Bulk Density (kg/dm3)	1.39
Topsoil Bulk Density (kg/dm3)	1.54
Topsoil Gravel Content (%)	9
Topsoil Organic Carbon (% weight)	0.86
Topsoil pH (H2O)	6.4
Topsoil CEC (clay) (cmol/kg)	38
Topsoil CEC (soil) (cmol/kg)	12
Topsoil Base Saturation (%)	89
Topsoil TEB (cmol/kg)	10
Topsoil Calcium Carbonate (% weight)	0
Topsoil Gypsum (% weight)	0
Topsoil Sodicity (ESP) (%)	1
Topsoil Salinity (ECe) (dS/m)	0.1
SUBSOIL (30-100 cm)	
Subsoil Sand Fraction (%)	39
Subsoil Silt Fraction (%)	27
Subsoil Clay Fraction (%)	34
Subsoil USDA Texture Classification	5
Subsoil Reference Bulk Density (kg/dm3)	1.33
Subsoil Bulk Density (kg/dm3)	1.52
Subsoil Gravel Content (%)	8
Subsoil Organic Carbon (% weight)	0.38
Subsoil pH (H2O)	6.3
Subsoil CEC (clay) (cmol/kg)	40
Subsoil CEC (soil) (cmol/kg)	15

Figure 6.3: HWSD database contents.

One of the handy tools is the soil water characteristics calculator developed by Keith Saxton from USDA Agricultural Research Service, shown on figure 6.4.

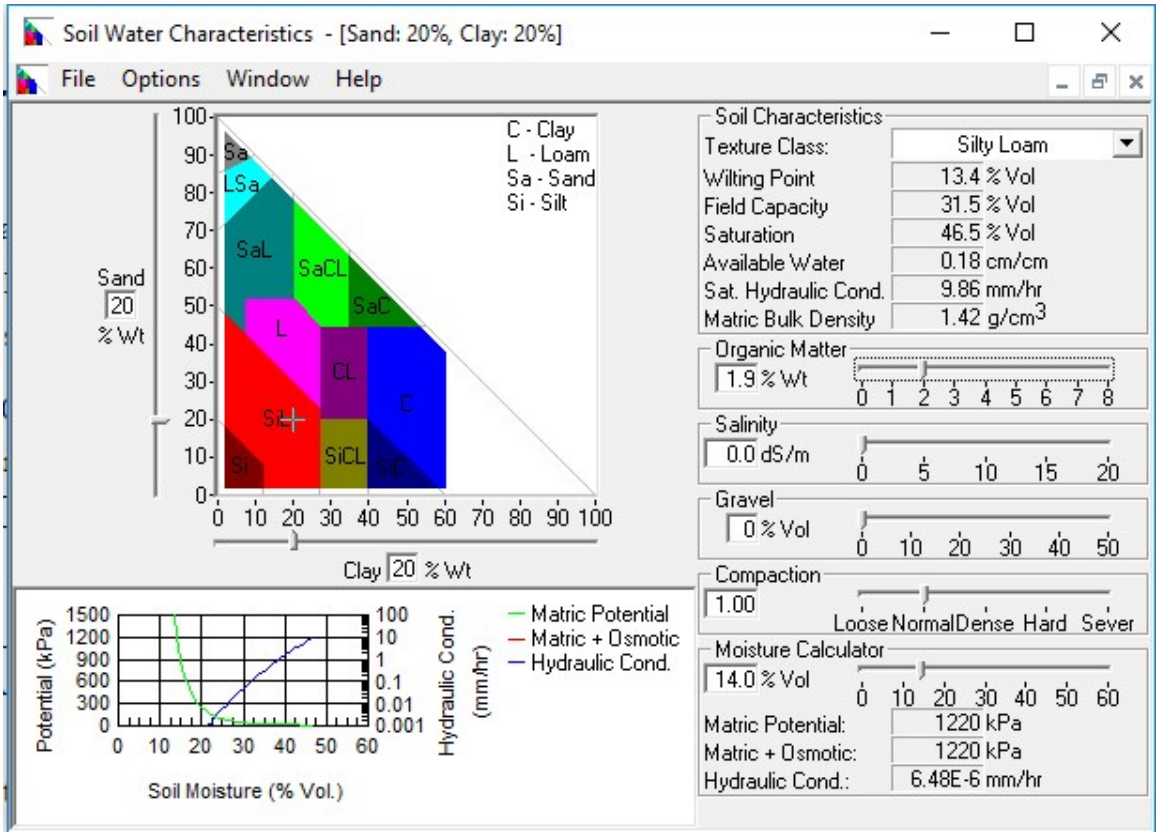


Figure 6.4: USDA Soil Water Characteristics Calculator user interface.

6.3 Simulations of Yields

Large number of simulations were run using DSSAT 4.7 and AquaCrop-GIS. Table 6.6 shows the crop models that were used in each crop yield model. Note that, AquaCrop-GIS uses AquaCrop v.4 crop model format, and is not compatible with crop files from v.6. Therefore, necessary modifications had to be made on v.6 crop files to make them work.

Fertilization and mulching were not considered in our simulations since these data were not available. The initial soil water level in AquaCrop is set to be at Field Capacity since it is a common practice to irrigate field with sufficient water before planting seeds. The planting date

Table 6.6: Crop models that were used in simulations in DSSAT and AquaCrop.

Crop	DSSAT Model	AquaCrop Model
Maize	McCurdy 84aa	MaizeGDD
Wheat	Default	WheatGDD
Soybean	COBB (8)	SoybeanGDD
Tomato	Sunny S-D 2010	TomatoGDD

for all simulations in DSSAT and AquaCrop was May 1st. Irrigation is set to be automatic when required. It is triggered by 80% of Field Capacity, and fill up to 100%. The efficiency was set to be 100%. In AquaCrop, the setup was exactly the same. 80% as MAD threshold, and fill back to Field Capacity. Water conservation in this project is not considered. The primary goal was to ensure crops get sufficient water to meet their needs.

For each country, yield results from all stations were collected, invalid ones were discarded. Invalid results mainly occured in DSSAT, where some experiments returned 0 yield, or under 20% of normal yield. Such problem only happened very few times in AquaCrop.

6.4 Machine Learning Based Model Fusion

Instead of considering only one model is correct and discarding predictions from other models, we have seen research works, in the area of Navigation and Control Technologies for Unmanned Systems, that treats each model as if they are sensor observations that can be combined or fused [78]. In other areas, such as chip design, Support Vector Regression (SVR) is used to replace complicated and time-consuming procedures like lithography simulation to improve design efficiency [79]. Although there are also a few studies involving machine learning techniques in predicting properties in agricultural products [80], main stream agricultural research is still largely relying on mechanistic models, and no model fusion works have been reported. As previous works have shown, individual model may not be very accurate without careful calibrations.

In the following context, we will introduce and compare 20 model fusion schemes that combine the raw outcomes from DSSAT and AquaCrop. Among these fusion schemes, 18 are based on machine learning techniques. A Neural Network is built, independent of DSSAT and AquaCrop

model, in order to study the possibility of replacing mechanistic crop yield model with faster machine learning models.

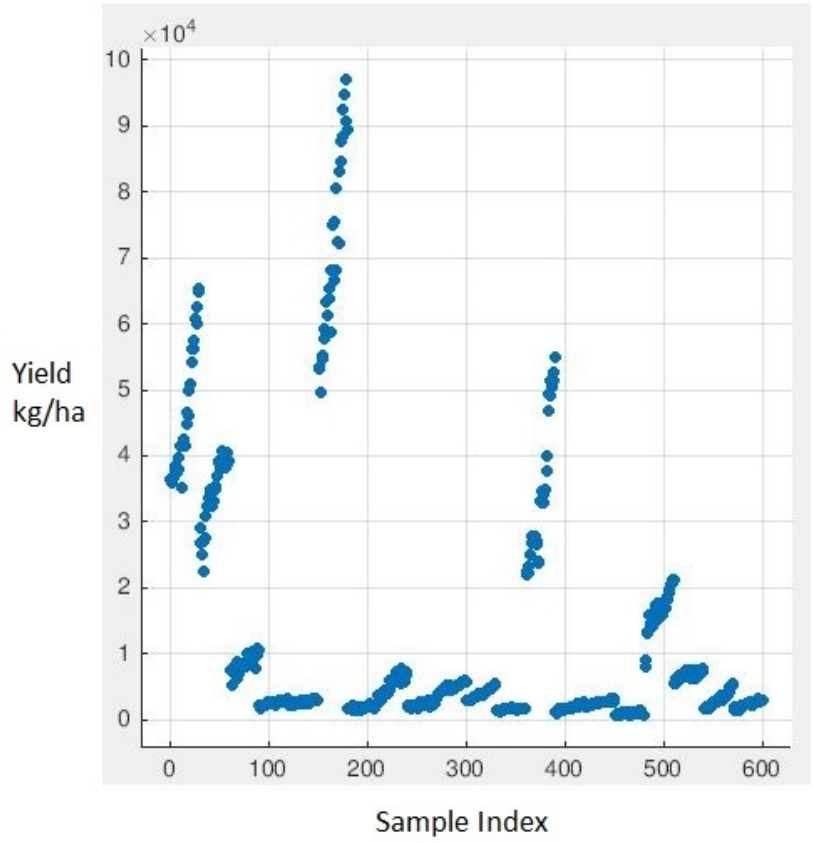


Figure 6.5: Actual crop yield data. Five top countries of each four crops over 30 years gives in total 600 samples.

The easiest ways to combine two models are taking the arithmetic and geometric means. If one compares figure 6.5 and figure 6.6, it is obvious that arithmetic mean does not count for an effective method. The predicted values are largely deviated from their targets. Here, the quality of any prediction is measured by R-squared, a statistical term that shows how close the data to the fitted regression line. R-squared is defined as: Explained variation/ Total variation. If the value is 100%, then all factors are accounted for, meaning we can generate 100% accurate prediction. Of course, such high performance seldom occurs.

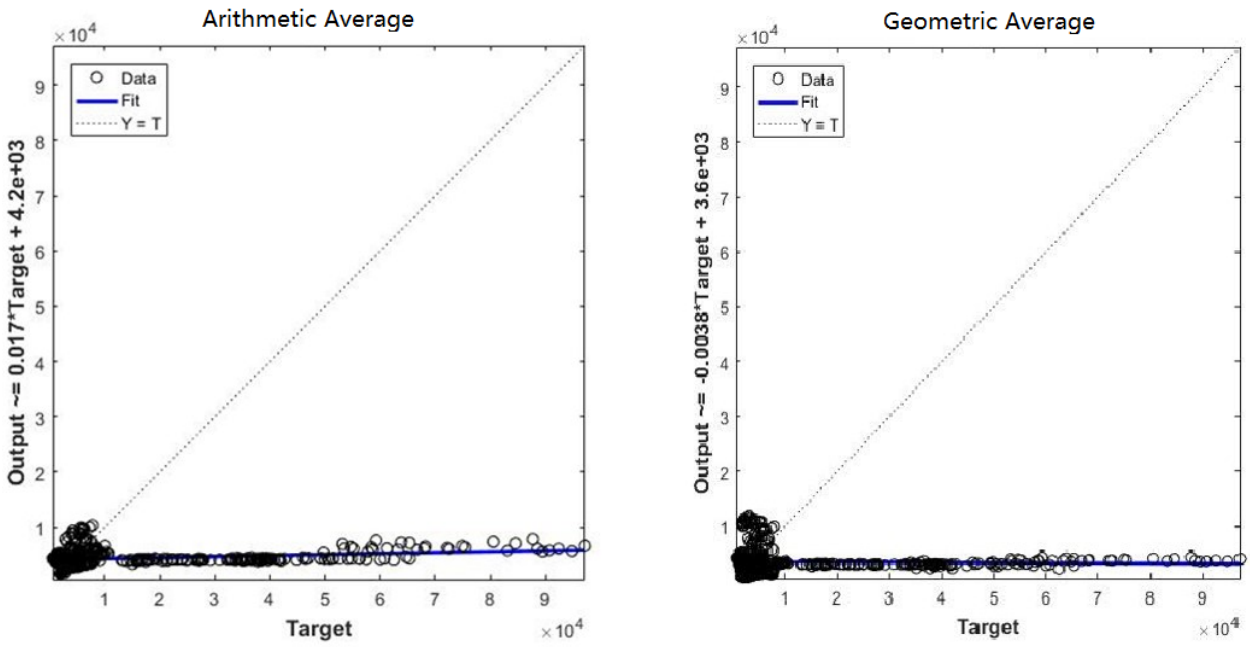


Figure 6.6: Regression of the arithmetic and geometric average of DSSAT and AquaCrop over the actual yield.

The geometric model is no better. As you can see in Figure 6.6, it presents a negative correlation between the predictions and actual yield. Above results are not surprising. In fact, they are consistent with the findings of Rising and Cane. Recall, in Table 6.1, the correlations between arithmetic mean and actual yield are between -0.15 to 0.35 for 5 crops.

Recall, in section 2, section 2, we discussed the mathematical expression of a Linear Regression model and the way to minimize the error using gradient descent. We constructed, beside the normal basic form of Linear Regression, three variations of Linear Regression: Interaction, Robust, and Stepwise.

For multivariate models, the hypothesis can be expressed as:

$$h_\theta(X) = X\theta \tag{6.18}$$

where $x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)}$ are n input features of a sample indexed by i.The Interaction Linear Re-

gression adds polynomial terms $\beta x_j^{(i)} x_k^{(i)}$ into the equation. Thus, it enhances the ability to handle complex relationship between predictor variables and response variable.

Remember, the cost function in normal linear regression model is given by:

$$J(\theta) = \frac{1}{2m}(X\theta - \overrightarrow{y})^T(X\theta - \overrightarrow{y}) \tag{6.19}$$

The shortcoming of using the summed squared error is that the estimates are extremely sensitive to the outliers. Therefore, an improvement was made to replace the original $J(\theta)$ with:

$$\sum_{i=1}^{n} \rho(\frac{y_i - \theta x_i}{\widehat{\sigma}}) \tag{6.20}$$

where $\rho()$ is a robust loss function, and σ is an error scale estimate. A common robust loss function is Huber's Ψ function [81].

Stepwise regression, as the name suggests, adds or removes a term according to its statistical significance in a regression at each step. The statistical significance is measured by statistical techniques such as F-tests and t-tests. The model will evolve as the iteration progresses.

In our case, Stepwise and Interaction type of linear regression outperform the normal and Robust Linear Regression. However, in general, linear regressions in their simple forms are less than ideal for crop yield prediction. The results are visualized on figure 6.7

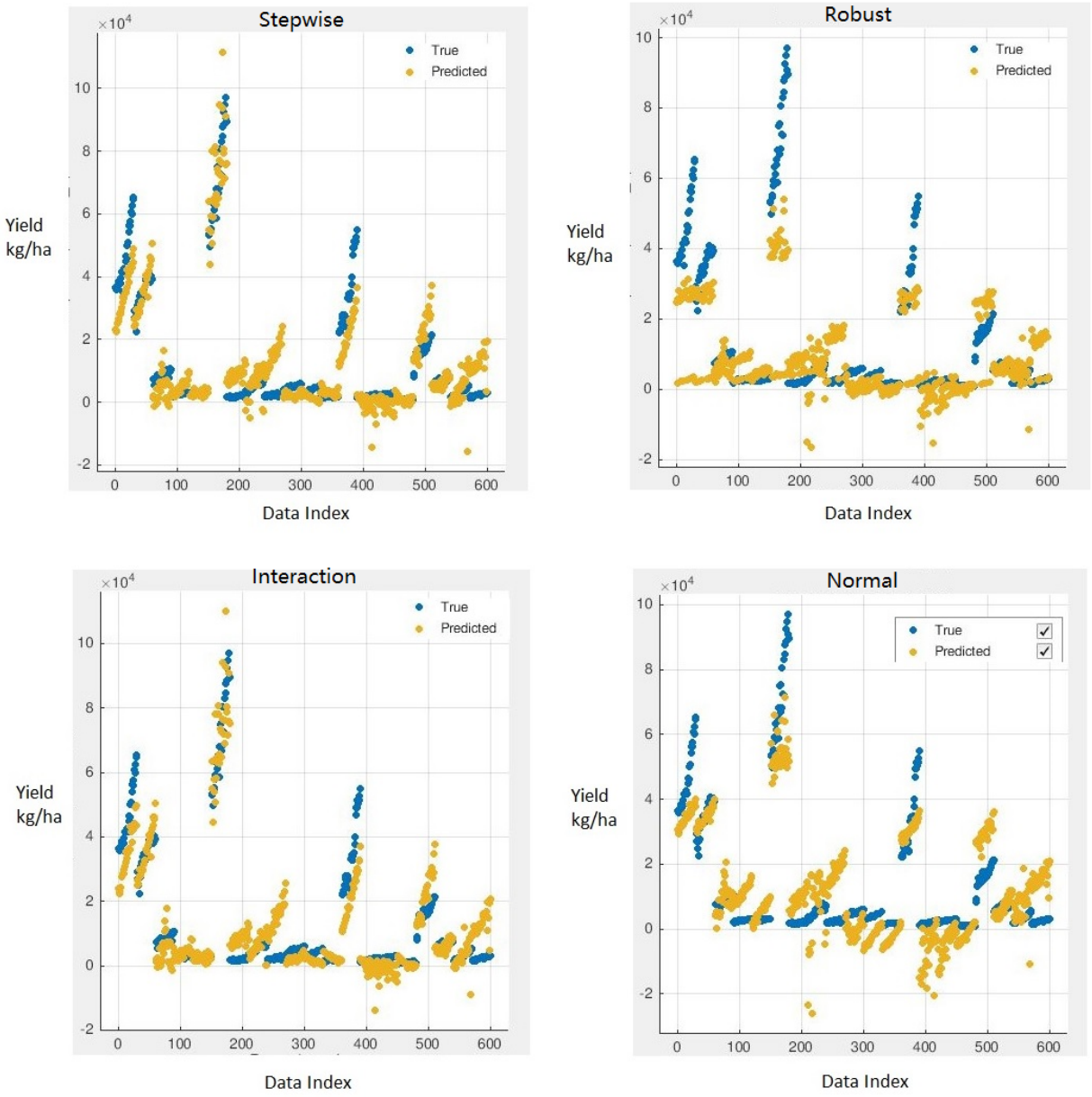


Figure 6.7: Regression plot of four Linear Regression models.

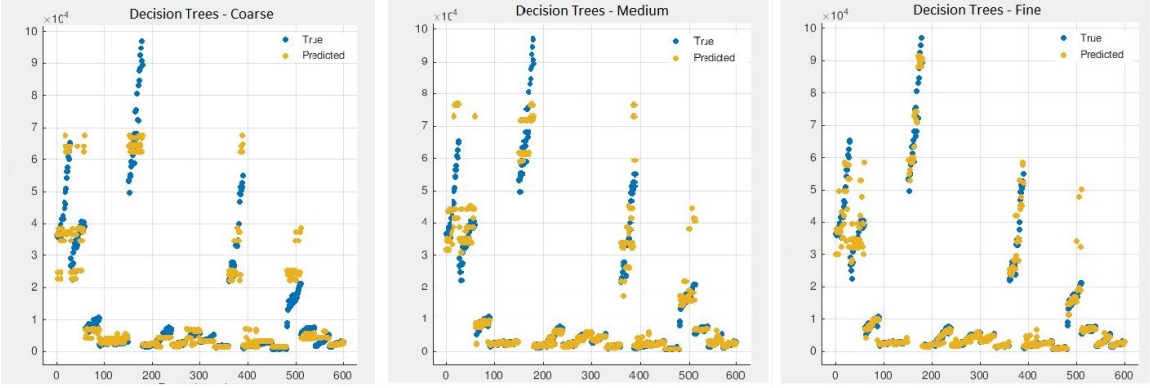


Figure 6.8: Regression plot of Coarse, Medium, and Fine Trees.

Tree models are better in terms of performance. Trees can be used in classification problems as well as regression. At each step, the algorithm chooses a point in the features space, and uses it as a reference to split the dataset by half. The objective is to find the best point to split so that the overall error is minimized.

$$\min_{\widehat{y}} \sum_{i:x_j^{(i)}>s}^{n} (\widehat{y} - y^{(i)})^2 + \min_{\widehat{y}} \sum_{i:x_j^{(i)}\leq s}^{n} (\widehat{y} - y^{(i)})^2 \tag{6.21}$$

where s is the splitting point. Taking the derivative of the summation w.r.t $y^{(i)}$, it is easy to find that $\widehat{y}$ should take the value of $\frac{1}{n}\sum_{i:x_j^{(i)}>s}^{n} y^{(i)}$

In tree structures, leaves are like unions of data points, which have labels attached to each data point. The splitting is repeated in an effort to differentiate data points with different label to the best ability. The process stops when the leaf is too small to be splitted. The smaller the leaf, the finer the tree is. In our case, we set the minimum leaf size to be 4, 12, and 36. From Figure 6.8, we can see that the overall performance of Trees are better than linear models. Moreover, Fine Trees seem to be well suit the job, with the highest R-squared value among all three Trees.

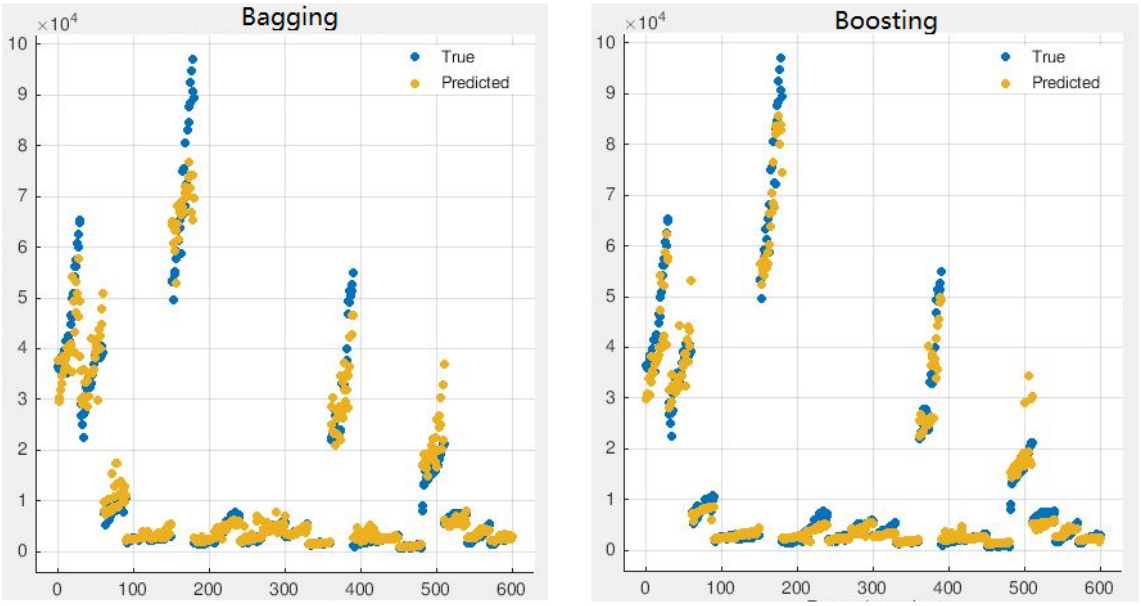


Figure 6.9: Regression plot of Ensemble Algorithms.

If you think that is good enough, then you have underestimated the power of machine learning. We go one step forward, concatenating Trees with ensemble techniques, and discover that the performance is further improved. Two of the ensemble techniques are implemented, Bagging and Boosting.

The basic idea behind Bagging is to create multiple models and train them with data drawn randomly from the entire training set. Because of the randomness, each subset is slightly different from each other. Therefore, models are trained at slightly different directions. Upon completion, the results are collected, and the final prediction is taken from the average. Such way, just like weather forecast, we can implant diversity at initialization and explore the possibility that might be ignored if there were only one model.

Boosting is an enhancement model that is built upon bagging. Rather than building n parallel models, in Boosting, models are built in series. Points that were not well predicted in previous model are subsequently given higher chance to be drawn from the training set for the next training.

Intuitively, and indeed proved by research and experiments, both bagging and boosting are

capable to suppress variance while providing high stability, thanks to the averaging. However, boosting tends to over-fit data since it may become too obsessed by those not well predicted data.

From the Figure 6.9, we can find that the Bagging algorithms provide as good as what Fine Trees can do. In addition, Boosting has boosted the performance even further.

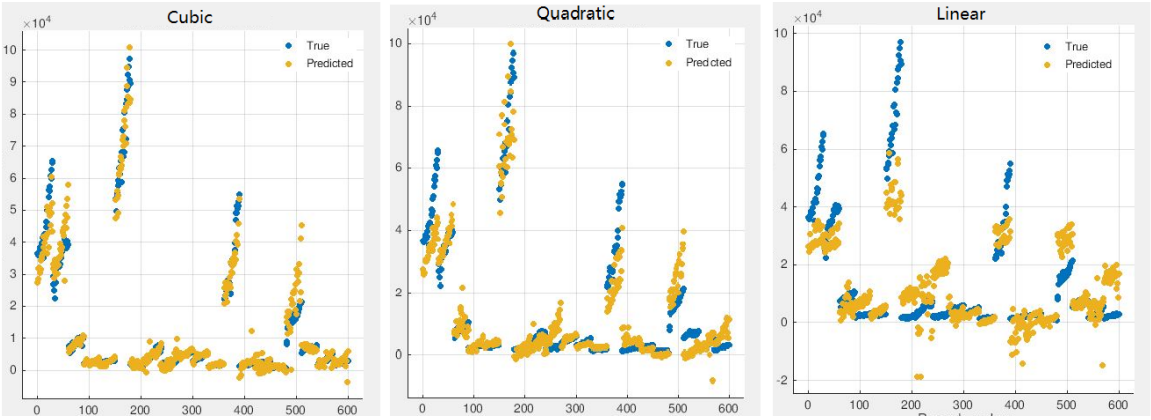


Figure 6.10: Regression plot of Support Vector Regression Algorithms.

In section 2, section 5, we have given detailed discussion in Support Vector Machine (SVM). By using different kernel functions, the SVMs can construct high dimensional contour line that separates different data very well. In our study, we constructed three SVMs based on linear, quadratic, and cubic kernels. Their performance varies largely from 0.67 to 0.97, as shown in Figure 6.10

Gaussian Processes Regression does not require a specified form of a function to describe observed data. Rather, it represents the function obliquely, but it can be very rigorous and powerful. Interested readers may find M. Ebden's introduction on Gaussian Process Regression (GPR) helpful [82]. Total four of different covariant functions are built. All of them demonstrate outstanding accuracy, with R-squared value above 99%. Their regression plots can be find in Figure 6.11

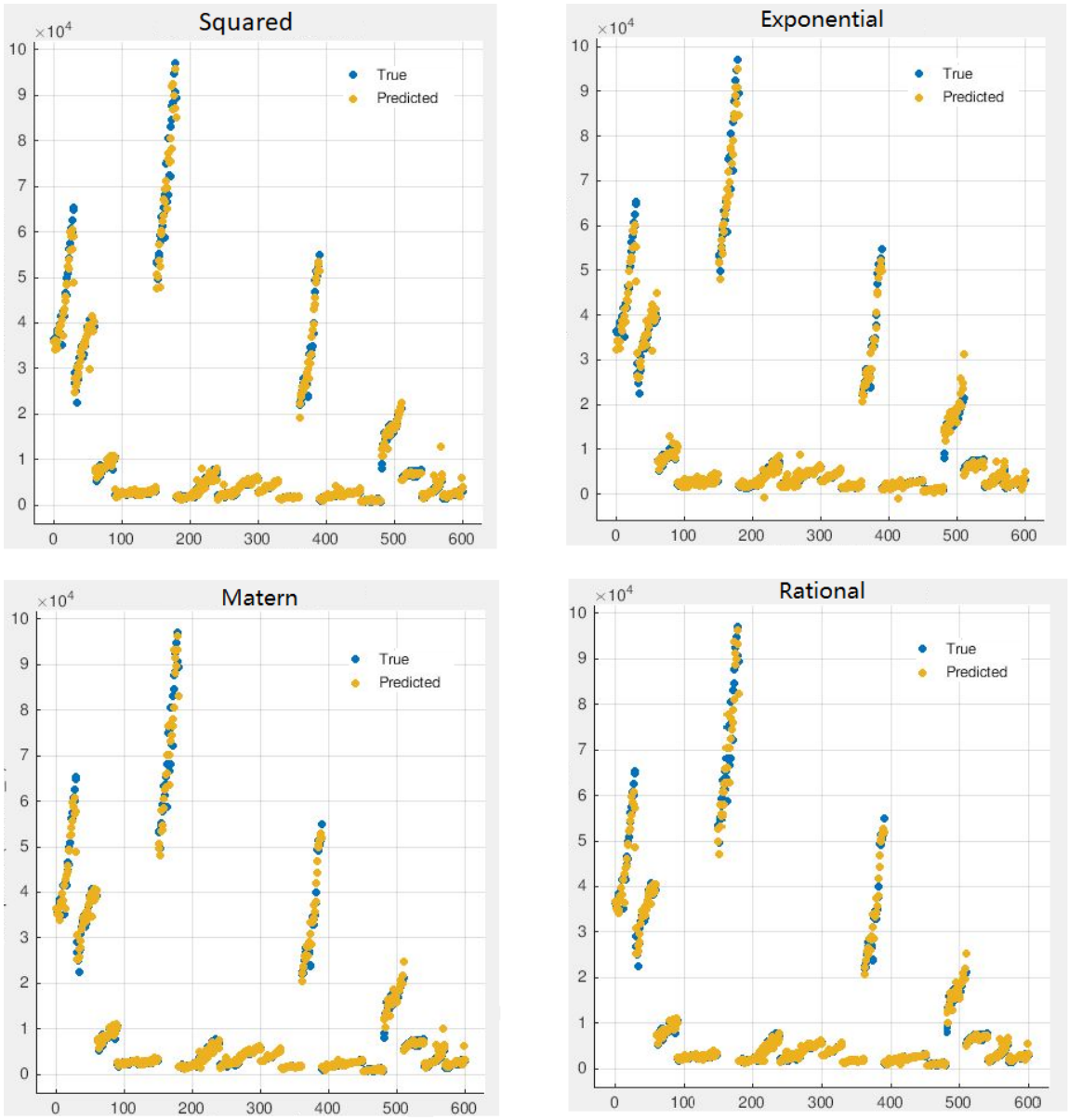


Figure 6.11: Regression plot of Gaussian Process Regression.

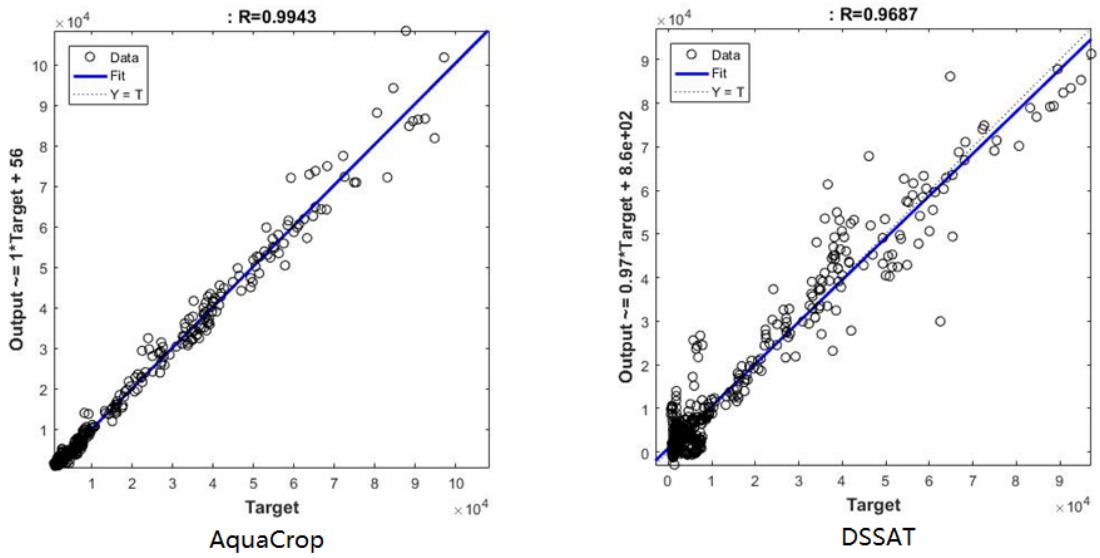


Figure 6.12: Neural Network Corrected Prediction based on DSSAT/AquaCrop + year + crop + country data.

Inspired by achievements made by other machine learning techniques, we wondered if it is possible to build a yield prediction Neural Network (NN) that takes in year, crop, country, and predictions from DSSAT and AquaCrop respectively, as inputs. Since the data coming out of the network becomes more correlated to the actual yield, we can concatenate the result with a SVM classifier which acts as a switch that only takes one of the "corrected prediction" out of the NN and use it as the ultimate prediction. The switch SVM learn from the absolute error between "corrected predictions" and actual yield, and decides when to switch on DSSAT and when to switch on AquaCrop. The training results of the front end NN are rather good in both DSSAT and AquaCrop cases as shown on figure 6.12, the final combined performance is presented on figure 6.13. The mechanism can be summarized as the following:

$$y_{final} = label \times y_{DSSAT} + (1 - label) \times y_{AquaCrop}; \tag{6.22}$$

$$label = SVM(year, crop, country) \subseteq [0, 1]; \tag{6.23}$$

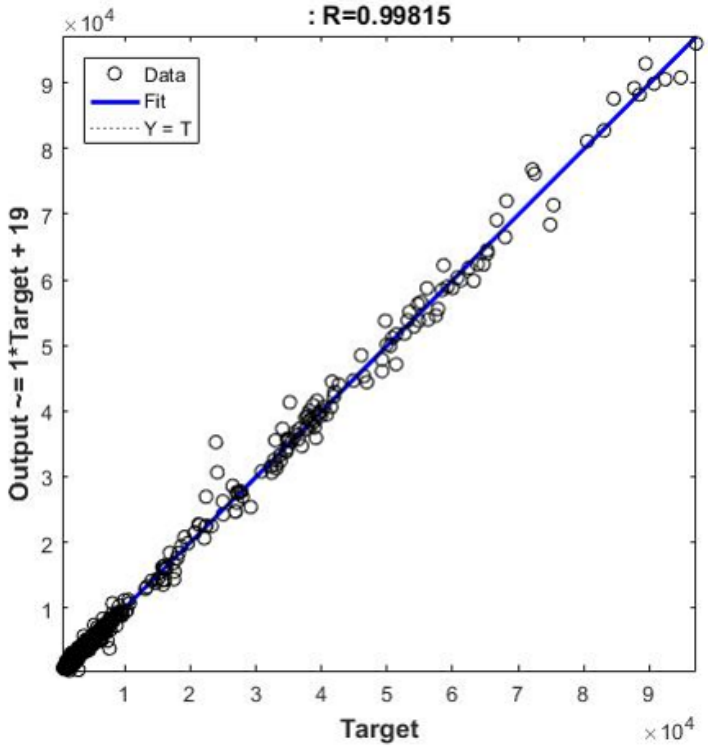


Figure 6.13: Regression plot of Neural Network concatenate SVM.

Lastly, we built a Neural Network that is independent of DSSAT or AquaCrop data. Year, crop, country, proportion of each soil type in a country, maximum, minimum, and mean latitude of the country are added to the feature space. The logic behind such design is this:

1. Crop growth is strongly correlated with energy/temperature, yet taking entire or part of the temperature or solar radiation data as input would significantly expanding the size of input feature space. Therefore, latitude becomes an ideal candidate since it is a constant, easily available, and is directly correlated with energy, more over, it is location specific.
2. The maximum, minimum, and mean latitudes provide references for expectation value and standard deviation.
3. Each type of crop has its own adaptability/yield response to different soil types. The proportion of each soil type in a country is estimated from selected stations in that country. Using

the proportion can eliminate the difference created by unequal number of experimental stations. Since our station selection rules are to pick locations as evenly as possible, the national wide crop response to soil should be have be well-modeled from these data.

The final result is surprisingly good. As shown in Figure 6.14, its R-squared value is the highest, and with smallest root-mean-square error (RMSE). The performance of all 21 models presented in this paper is summarized in Table 6.7

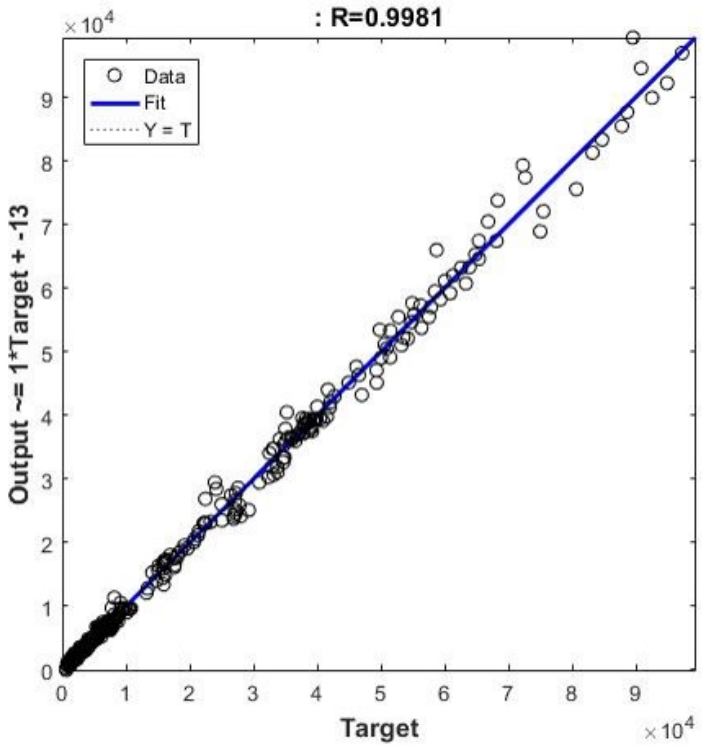


Figure 6.14: Regression plot of a model free independent Neural Network.

6.5 Conclusion

In this section, we have simulated four major agricultural crops in twenty countries over thirty years, using DSSAT and AquaCrop crop yield models. Previous works and our data have shown that without calibration, outcomes from professional crop yield models may deviate from actual

Table 6.7: Machine Learning techniques used in model fusion and their performance.

Method	**R-Squared**	**RMSE**
Arithmetic Average	0.24	20713
Geometric Average	-0.03	21443
Linear Regression	0.74	9897
Interactions Linear Regression	0.87	7066.7
Robust Linear Regression	0.39	14969
Stepwise Linear Regression	0.86	7132
Decision Trees - Fine	0.96	3672.3
Decision Trees - Medium	0.93	5154.1
Decision Trees - Coarse	0.9	6170.9
Ensemble - Boosted Trees	0.97	3196.9
Ensemble - Bagged Trees	0.95	4210.9
SVM - Linear Kernel	0.67	11080
SVM - Quadratic Kernel	0.89	6236.6
SVM - Cubic Kernel	0.96	3842.8
Squared Exponential Gaussian Process Regression	0.99	1769.3
Matern Gaussian Process Regression	0.99	1623.2
Exponential Gaussian Process Regression	0.99	1837.7
Rational Quadratic Gaussian Process Regression	0.99	1636.8
Neural Networks + SVM correction	0.99	1243
Linear Regression on Neural Network corrected data	0.99	1430
Model Free Independent Neural Networks	0.99	1185

yield significantly at country level. We implemented 20 model fusion schemes, from the easiest linear model to the very complicated multi-layered machine learning models. The prediction qualities measured by the R-square value and Root-mean-squared-error are steadily improved. Among these models, various kinds of Gaussian Process Regression models and schemes based on Neural Networks have achieved very desirable results. In addition, we proposed an independent Neural Network that is trained without including DSSAT nor AquaCrop data. Such Network uses "year" to catch variation in time domain, "latitude" to model solar energy, and an estimation of soil types distribution in a country to estimate country-wide crop-soil response. The outcome ranks the most accurate prediction in the record. Its success clearly demonstrates the great potential of machine learning techniques in agricultural research, and the possibility of replacing traditional rigorous but time-consuming professional crop yield models for large-scale (coarse) applications.

7. SUMMARY AND CONCLUSIONS

Benefiting from recent development of wireless communication technologies and artificial intelligence, the efficiency and productivity of traditional industries like power and agriculture can be significantly improved. This dissertation focuses on modeling and control techniques that are used in various smart systems.

Accurate predictions of energy consumption at both household level and community level help electricity companies to plan production and integration of renewable energy more efficiently. In addition, they are also valuable references for detecting power theft and cyber-attacks. A comparative study on accurate predictive method for energy consumption is presented. Different Neural Network (NN), including conventional NN, Deep Neural Networks (DNN), and Sliding Window Neural Networks (SWNN), are compared. SWNN uses a window of historical data to predict the future energy consumption. Our experimental study shows that the conventional NN can achieve high accuracy in prediction while deep NN does not generate better results. Through data normalization and temporal relationship exploration, SWNN becomes superior to conventional methods and achieves above 99.5% accuracy with a more condensed error distribution.

An advanced automated irrigation system is designed and built with high water-use efficiency. In its kernel, a reinforcement learning based irrigation control algorithm enables the system maintaining optimal soil water level at any given crop growth stage. The delayed reward of crop yield is handled by the temporal difference technique. The learning process can be based on both off-line simulation and real data from sensors and crop yield. Neural network based fast models for soil water level and crop yield are developed to improve the scalability of learning. Simulations for various geographic locations and crop types show that the proposed method can significantly increase net return considering both crop yield and water expense. A well-designed web-based user interface provides remote controllability of the system. Well-informed decision-making is guaranteed by real-time soil moisture and weather forecast data. The former is collected through local wireless sensor network, and the later is downloaded from NNDC Climate database [76] . An outstanding

operational precision is made possible by the combination of GPS, wireless communication, and PID speed control.

Lastly, model fusion techniques are studied and compared. Crop yield models play important roles in agriculture research. In particular, the effectiveness of our reinforcement learning control algorithm depends on simulation data generated from these models. 30 years of soil texture, historical weather, and harvest data of over 200 stations in 9 countries are collected. Simulations are run on both DSSAT and AquaCrop models. R-square value and root-mean-squared-error are measured from 20 model fusion schemes, from the easiest linear model to the very complicated multi-layered machine learning models. Among these models, various kinds of Gaussian Process Regression models and schemes based on Neural Networks have achieved very desirable results. In addition, an independent Neural Network that is trained without including DSSAT or AquaCrop data is proposed. Such Network uses "year" to catch variation in time domain, "latitude" to model solar energy, and an estimation of soil type distribution in a country to estimate country-wide crop-soil response. This study provides viable ways to combine multiple crop yield models, and demonstrates the great potential of machine learning techniques in agricultural research.

REFERENCES

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[2] "Atmel 8bit micrcontroller in system programmable flash datasheet." http://ww1.microchip.com/downloads/en/DeviceDoc, 2015. [Online; accessed 05/01/2018].

[3] "Intel Edison compute module hardware guide." https://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison-module_HG_331189.pdf, 2016. [Online; accessed 05/01/2018].

[4] "MSP430x2xx family user guide." http://www.ti.com/lit/ug/slau144j/slau144j.pdf, 2013. [Online; accessed 05/01/2018].

[5] Suryakanta, "How to identifiy silt and clay in the field?." http://http://civilblog.org/2014/02/10/how-to-identify-silt-clay-in-the-field/, 2014. [Online; accessed 05/15/2018].

[6] R. A and Garcia-Gaines, "USCS and the USDA soil classification system." https://http://www.dtic.mil/dtic/tr/fulltext/u2/a614144.pdf, 2015. [Online; accessed 05/01/2018].

[7] D. Raes, P. Steduto, and T. Hsiao, "AquaCrop version 6.0 reference manual." http://www.fao.org/3/a-br246e.pdf, 2017. [Online; accessed 05/01/2018].

[8] C. Brouwer, "Irrigation Water Management: Training Manual No. 1 - Introduction to Irrigation." http://www.fao.org/docrep/r4082e/r4082e00.htm, 1985. [Online; accessed 05/01/2018].

[9] R. A. Garcia-Gaines, "Soil water terminology handout." http://www.irrigation.org/IA/FileUploads/IA/Certification/SoilWaterTerminology.pdf, 2017. [Online; accessed 05/01/2018].

[10] NOAA, "Ensemble prediction systems." http://www.wpc.ncep.noaa.gov/ensembletraining/, 2006. [Online; accessed 05/01/2018].

[11] J. Murphy, "What's the Difference Between AC Induction, Permanent Magnet, and Servomotor Technologies." http://www.machinedesign.com/motorsdrives/, 2012. [Online; accessed 05/11/2018].

[12] "PID controller." https://en.wikipedia.org/wiki/PID_controller, 2017. [Online; accessed 05/15/2018].

[13] R. Allen, L. Pereira, D. Raes, and M. Smith, *Crop evapotranspiration*, p. 56. Boston, MA: FAO, 1998.

[14] *The nature and properties of soils (9th ed.)*. Collier Macmillan, 1984.

[15] J. Rising and M. Cane, "Comparison of global agricultural modeling results." http://www.existencia.org/pro/wp-content/uploads/2011/05/paper2.pdf, 2011. [Online; accessed 05/01/2018].

[16] Food and A. O. of the United Nations, "FAOSTAT, Crops." http://www.fao.org/faostat/en/#data, 2018. [Online; accessed 05/15/2018].

[17] K. Worden, W. A. Bullough, and J. Haywood, *Smart Technologies*. World Scientific Publishing Co. Pte. Ltd, 2003.

[18] J. Jones, G. Hoogenboom, C. Porter, K. Boote, W. Batchelor, L. Hunt, P. Wilkens, U. Singh, A. Gijsman, and J. Ritchie, "The DSSAT cropping system model," *European Journal of Agronomy*, vol. 18, no. 3, pp. 235 – 265, 2003. Modelling Cropping Systems: Science, Software and Applications.

[19] I. W. An, "Renewable energy in the EU." http://http://ec.europa.eu/eurostat/documents/2995521/8612324/8-25012018-AP-EN.pdf/9d28caef-1961-4dd1-a901-af18f121fb2d, 2018. [Online; accessed 05/01/2018].

[20] C. Murphy-Levesque and C. Donnan, *IEA Wind TCP 2016 Annual Report*, p. 64. Boulder, Colorado 80303: PWT Communications, LLC, 2016.

[21] M. Rosas-Casals, S. Valverde, and R. V. Sole, "Topological vulnerability of the European power grid under errors and attacks," *International Journal of Bifurcation and Chaos*, vol. 17, no. 07, pp. 2465–2475, 2007.

[22] B. Schneier, "Lessons from the Dyn DDoS attack." https://www.schneier.com/blog/archives/2016/11/lessons_from_th_5.html, 2014. [Online; accessed 05/01/2018].

[23] "Final report on the august 14, 2003 blackout in the United States and Canada.." https://reports.energy.gov/B-F-Web-Part1.pdf, 2004. [Online; accessed 05/01/2018].

[24] Y. Liu, M. K. Reiter, and P. Ning, "False data injection attacks against state estimation in electric power grids," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, vol. 14, pp. 13:1–13:33, 2009.

[25] M. D. Dukes, "Water conservation potential of landscape irrigation smart controllers," *Transactions of the ASABE*, vol. 55, no. 02, pp. 563–569, 2012.

[26] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5 – 10, 2004.

[27] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing, "Utilizing green energy prediction to schedule mixed batch and service jobs in data centers," in *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, HotPower '11, (New York, NY, USA), pp. 5:1–5:5, ACM, 2011.

[28] A. Kimbara, S. Kurosu, R. Endo, K. Kamimura, T. Matsuba, and A. Yamada, "On-line prediction for load profile of an air-conditioning system," vol. 101, pp. 198–207, 1995. cited By 21.

[29] M. Anstett and J. Kreider, "Application of neural networking models to predict energy use," vol. 99, pp. 505–517, 1993. cited By 46.

[30] J. Yang, H. Rivard, and R. Zmeureanu, "On-line building energy prediction using adaptive artificial neural networks," *Energy and Buildings*, vol. 37, no. 12, pp. 1250 – 1259, 2005.

[31] A. C. McCarthy, N. H. Hancock, and S. R. Raine, "Simulation of irrigation control strategies for cotton using model predictive control within the VARIwise simulation framework," *Computers and Electronics in Agriculture*, vol. 101, pp. 135 – 147, 2014.

[32] N. SchÃijtze and G. Schmitz, "Neuro-dynamic programming as a new framework for decision support for deficit irrigation sytems," 01 2007.

[33] Maryt, "Intel Edison discontinuation notice." https://communities.intel.com/docs/DOC-112093, 2017.

[34] "libmraa - low level skeleton library for communication on GNU/Linux platforms." https://github.com/intel-iot-devkit/mraa, 2015. [Online; accessed 05/15/2018].

[35] J. L. Merriam, "A management control concept for determining the economical depth and frequency of irrigation," in *Transactions of the ASAE*.

[36] "Definition of crop." https://www.merriam-webster.com/dictionary/crop, 2017. [Online; accessed 05/01/2018].

[37] L. G. Swan and V. I. Ugursal, "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 8, pp. 1819 – 1835, 2009.

[38] B. M. Larsen and R. Nesbakken, "Household electricity end-use consumption: results from econometric and engineering models," *Energy Economics*, vol. 26, no. 2, pp. 179 – 200, 2004.

[39] M. Aydinalp, V. I. Ugursal, and A. S. Fung, "Modeling of the appliance, lighting, and space-cooling energy consumptions in the residential sector using neural networks," *Applied Energy*, vol. 71, no. 2, pp. 87 – 110, 2002.

[40] H. xiang Zhao and F. MagoulÃĺs, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586 – 3592, 2012.

[41] G. K. Tso and K. K. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761 – 1768, 2007.

[42] B. B. Ekici and U. T. Aksoy, "Prediction of building energy consumption by using artificial neural networks," *Advances in Engineering Software*, vol. 40, no. 5, pp. 356 – 362, 2009.

[43] D. C. Park, M. A. El-Sharkawi, R. J. Marks, L. E. Atlas, and M. J. Damborg, "Electric load forecasting using an artificial neural network," *IEEE Transactions on Power Systems*, vol. 6, pp. 442–449, May 1991.

[44] T. M. Peng, N. F. Hubele, and G. G. Karady, "Advancement in the application of neural networks for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 7, pp. 250–257, Feb 1992.

[45] S.-C. Wang, *Artificial Neural Network*, pp. 81–100. Boston, MA: Springer US, 2003.

[46] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *International 1989 Joint Conference on Neural Networks*, pp. 593–605 vol.1, 1989.

[47] R. Rojas, *Neural Networks*. Springer-Verlag Berlin Heidelberg, 1996.

[48] H. Demuth, M. Beale, and M. Hagan, *Neural Network Toolbox 6 User's Guide*. The MathWorks, Inc, 2009.

[49] "Belgian wind farm data." http://www.elia.be/en/grid-data/powergeneration/wind-power, 01 2014. [Online; accessed 10/01/2016].

[50] L. Liu, Y. Liu, L. Wang, A. Zomaya, and S. Hu, "Economical and balanced energy usage in the smart home infrastructure: A tutorial and new results," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, pp. 556–570, Dec 2015.

[51] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system

for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, (Savannah, GA), pp. 265–283, USENIX Association, 2016.

[52] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *International 1989 Joint Conference on Neural Networks*, pp. 593–605 vol.1, 1989.

[53] L. M. Saini and M. K. Soni, "Artificial neural network based peak load forecasting using Levenberg-Marquardt and quasi-Newton methods," *IEE Proceedings - Generation, Transmission and Distribution*, vol. 149, pp. 578–584, Sep 2002.

[54] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. MIT Press, 1998.

[55] M. E. Jensen, "Scheduling irrigation with computers," *Journal of Soil and Water Conservation*, vol. 24, no. 5, pp. 193 – 195, 1969.

[56] B. Ghahraman, , and A. R. Sepaskhah, "Use of a water deficit sensitivity index for partial irrigation scheduling of wheat and barley," *Irrigation Science*, vol. 18, pp. 11–16, Nov 1997.

[57] T. A. Howell, *Optimization of grain sorghum water use efficiency under high frequency irrigation by system simulation and stochastic dynamic programming*. PhD thesis, Texas A&M University, 1974.

[58] J. D. Sunantara and J. Ramirez, "Optimal stochastic multicrop seasonal and intraseasonal irrigation control," *Journal of Water Resources Planning and Management-asce - J WATER RESOUR PLAN MAN-ASCE*, vol. 123, 01 1997.

[59] R. Wardlaw and K. Bhaktikul, "Application of genetic algorithms for irrigation water scheduling," *Irrigation and Drainage*, vol. 53, pp. 397–414, June 2004.

[60] K. L. Moore and Y. Chen, "Iterative learning control approach to a diffusion control problem in an irrigation application," in *2006 International Conference on Mechatronics and Automation*, pp. 1329–1334, June 2006.

[61] Y. Kim, R. G. Evans, and W. M. Iversen, "Remote sensing and control of an irrigation system using a distributed wireless sensor network," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, pp. 1379–1387, July 2008.

[62] Y. Kim, R. Evans, and W. Iversen, "Evaluation of closed-loop site-specific irrigation with wireless sensor network," vol. 135, 02 2009.

[63] F. Capraro, D. Patino, S. Tosetti, and C. Schugurensky, "Neural network-based irrigation control for precision agriculture," in *2008 IEEE International Conference on Networking, Sensing and Control*, pp. 357–362, April 2008.

[64] S.Shaughnessy, S. R. Evett, P. D. Colaizzi, and T. A. Howell, "ASABE applied engineering in agriculture," in *2008 IEEE International Conference on Networking, Sensing and Control*, vol. 29, pp. 853–864, Nov. 2013.

[65] S.Shaughnessy, S. R. Evett, P. D. Colaizzi, and T. A. Howell, "Transactions of the asabe," in *2008 IEEE International Conference on Networking, Sensing and Control*, vol. 55, pp. 451–461, 2012.

[66] R. Romero, J. Muriel, I. Garcia, and D. M. de la Pena, "Research on automatic irrigation control: State of the art and recent results," *Agricultural Water Management*, vol. 114, pp. 59 – 66, 2012. For a better use and distribution of water.

[67] A. C. McCarthy, N. H. Hancock, and S. R. Raine, "Advanced process control of irrigation: the current state and an analysis to aid future development," *Irrigation Science*, vol. 31, pp. 183–192, May 2013.

[68] R. G. Evans, J. LaRue, K. C. Stone, and B. A. King, "Adoption of site-specific variable rate sprinkler irrigation systems," *Irrigation Science*, vol. 31, pp. 871–887, Jul 2013.

[69] "Corn price today." http://markets.businessinsider.com/commodities/corn-price, 2017. [Online; accessed 06/30/2017].

[70] K. Skaarhoj, "Food price monitoring and analysis." http://www.fao.org/giews/food-prices/international-prices/detail/en/c/895692/, 2017. [Online; accessed 07/11/2017].

[71] D. Wichelns, "Agricultural water pricing: United States." https://www.oecd.org/unitedstates/45016437.pdf, 2010. [Online; accessed 07/11/2017].

[72] J. J. More, ed., *The Levenberg-Marquardt algorithm: implementation and theory*.

[73] Y. Jame and H. Cutforth, "Crop growth models for decision support systems," *Journal of Plant Science*, pp. 9–19, 1996.

[74] W. Schlenker and M. J. Roberts, "Nonlinear temperature effects indicate severe damages to U.S. crop yields under climate change," *Proceedings of the National Academy of Sciences*, vol. 106, no. 37, pp. 15594–15598, 2009.

[75] L.Sun, Y.Yang, J.Hu, D.Porter, T.Marek, and C.Hillyer, "Reinforcement learning control for water- efficient agricultural irrigation," *Proceedings of 16th IEEE International Conference on Ubiquitous Computing and Communications*, 2017.

[76] "NNDC climate data online." https://www7.ncdc.noaa.gov/CDO/cdo, 2018. [Online; accessed 05/15/2018].

[77] C. F. Bohren and D. Huffman, *Absorption and Scattering of Light by Small Particles*, pp. 123–126. Wiley, 2008.

[78] S. J. Julier and H. F. Durrant-Whyte, "Horizontal model fusion paradigm," *Proc.SPIE*, vol. 2738, pp. 2738 – 2738 – 12, 1996.

[79] T. Kimura, T. Matsunawa, S. Nojima, and D. Z. Pan, "Hybrid hotspot detection using regression model and SOCS kernels," *PIE Intl. Symp. Advanced Lithography Conference*, pp. 21–25, 2016.

[80] S. Wang, M. Huang, and Q. Zhu, "Model fusion for prediction of apple firmness using hyperspectral scattering image," *Computers and Electronics in Agriculture*, vol. 80, pp. 1 – 7, 2012.

[81] C. Yu and W. Yao, "Robust linear regression: A review and comparison," *Communications in Statistics - Simulation and Computation*, vol. 46, no. 8, pp. 6261–6282, 2017.

[82] C. E. Rasmussen, *Gaussian Processes in Machine Learning*, pp. 63–71. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.