CA2 – Food and Workout Recommendation Intelligent Self-Learning System

18-Nov-2019 | Version 1.0

Team 6

Alfred Tay Wenjie
Wang Zilong
Wong Yoke Keong

Prepared for NUS-ISS Master of Technology in Intelligent Systems ISY5005 Intelligent Software Agents (Self-Learning Systems) Group Project

Table of Contents

1. Introduction

1.1 Problem Statement

In today's fast-paced society, it is very challenging to eat right and exercise right. Most people know the benefits and desire to eat a balanced diet and to do the right amount of exercise, but many just do not have the time to plan what to eat and how much and what exercise to do. Some tired to count calories but there is up to now, no quick and accurate way to do so.

On the other hand, eating is also an activity which can bring happiness. We all feel happy when we are served our favourite food. How do we maximise this happiness and at the same time avoid putting on extra weight which makes us feel guilty the next time we ordered that chocolate ice-cream?

1.2 Proposed Solution

The proposed solution is an Intelligent Self-Learning System which recommends to the user a highly personalised daily eating and exercise plan which keeps the calories in check and maximises the user's happiness from food consumption. The exercise plan is also being optimised such that it is just enough to burn away the input calories by an amount based on user's weight loss goal.

2. Dataset

Two datasets are used in this project. One is a listing of food items with its respective calories, utility and carbohydrate, fat and protein composite. The calories and carbohydrate, fat and protein composite information is taken from https://www.myfitnesspal.com/food/search. The utility value is arbitrarily populated for the development of the system. In a production setting, these values will be elicited from the user.

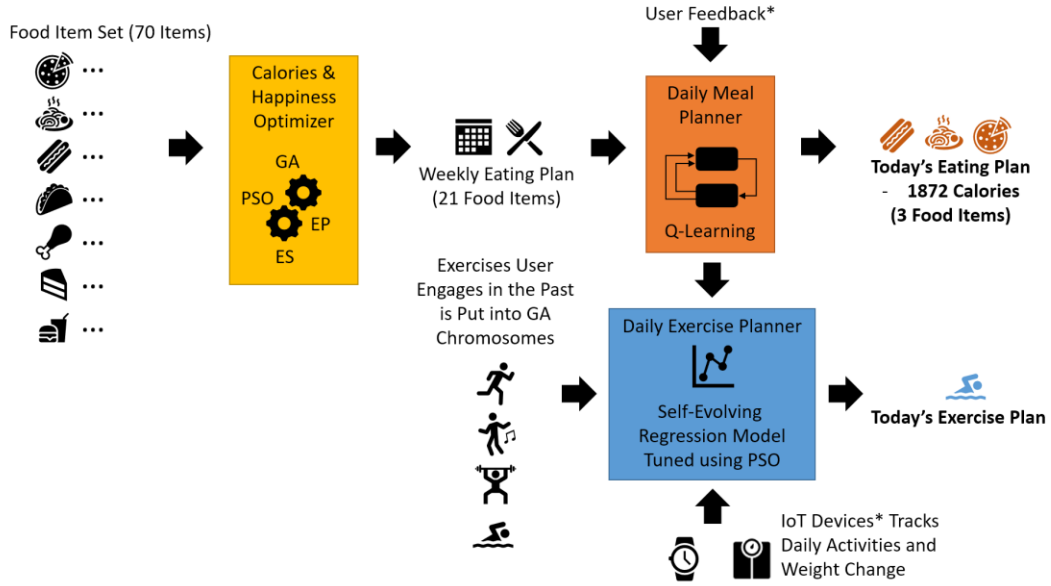
	Food	Calories	Utility	Cabohydrate	Fat	Protein
0	Murtabak	674	60	0.18	0.75	0.07
1	Ee Mee Seafood (Fried)	1010	90	0.31	0.54	0.15
2	Nasi Goreng Sayur	920	50	0.67	0.27	0.06
3	Claypot Rice	896	80	0.41	0.37	0.21
4	Ban Mian	476	50	0.41	0.42	0.18

The second dataset is provided by ChrisBow on Kaggle. This dataset contains information like calorie intake, intake and whether the author went for exercises like brisk walks and runs. For this assignment, only 4 variables 'calories', 'walk' (representing if brisk walk of at least 20 minutes has been completed), 'run' (representing if a run of 2.5 miles has been completed) and 'change' (weight change in ounces, target variable) are used. In future, rather than just indicator values for exercises, the observations can be continuous-value data from fitness trackers like Fitbit and Garmin, providing deeper data insights like minutes spent.

	Date	Stone	Pounds	Ounces	weight_oz	calories	cals_per_oz	five_donuts	walk	run	wine	prot	weight	change
0	7/30/2018	12.0	2.0	6.0	2726.0	1950.0	0.72	1.0	1.0	0.0	0.0	0.0	0.0	-30.0
1	7/31/2018	12.0	0.0	8.0	2696.0	2600.0	0.96	1.0	0.0	0.0	0.0	0.0	0.0	8.0
2	8/1/2018	12.0	1.0	0.0	2704.0	2500.0	0.92	1.0	1.0	0.0	0.0	0.0	0.0	0.0
3	8/2/2018	12.0	1.0	0.0	2704.0	1850.0	0.68	1.0	1.0	0.0	1.0	0.0	0.0	-40.0
4	8/3/2018	11.0	12.0	8.0	2664.0	2900.0	1.09	1.0	1.0	0.0	0.0	0.0	0.0	14.0

3. Solution Overview

The solution comprises (1) Calories and Happiness Optimizer, (2) Daily Meal Planner and (3) Daily Exercise Planner.



The Calories and Happiness Optimizer selects 21 food items from the food listing comprising of 70 initial food item set. Each food item has a utility score elicited from the user representing how much the user likes that food item. The higher the utility score, the happier the user will feel after consuming the food item. The goal of the optimizer is to maximise the total utility score (thus maximising happiness) and to keep within a specific daily calorie intake. The 21 food items represent the eating plan for a week and must not have repeated food items. Various Evolutionary Computation techniques are used to perform the optimization for comparison. From week to week, a rule can be imposed such that not more than 10 food items can be repeated.

The Daily Meal Planner takes the weekly eating plan and uses Q-Learning to learn the user's preference on what food they like or do not like to eat in close succession. An example is that if the user ate chicken rice during lunch, he may enjoy duck rice less for dinner as compared to if he had laksa for lunch. The output of the Daily Meal Planner is an optimised sequence for the 21 food items organised into 7 x 3 meals a day.

The Daily Exercise Planner relies on a Calories Output Regression Model to recommend the exercise(s) for the day. It uses Differential Evolution algorithm to search for the most suitable exercise(s) to offset the calories consumed based on the daily eating plan. The list of possible exercises for recommendation comes from a historical list of exercises which the user has engaged in the past. This data will be collected through a smart watch which tracks user's daily exercise activities. The Calories Output Regression Model is a Neural Network to be trained using data from the smart watch and smart weighting scale. The Neural Network is capable of self-evolving when given new input training data and the weight of the Neural Network is tuned using PSO.

*The User Feedback and IoT Device integration will not be covered in the scope of this CA and is included in the diagram and in the preceding paragraph to show the end-to-end data flow of the production system. The main system components have been developed such that when in production, the system is capable of continuous self-learning and self-evolving according to the changing environment such as changing user food preference, new food items and new exercises.

4. Calories and Happiness Optimizer

Evolutionary Computation techniques were used to select 21 food items from 70 food items to maximise utility and keep within a target daily calories intake. Four different techniques, namely Genetic Algorithm, Evolutionary Programming, Evolution Strategies and Particle Swarm Optimization, were experimented with to find the best performing technique for this task.

4.1 Genetic Algorithm

Genetic algorithm is suitable for this task because the task is a combinatorial optimization problem with a maximum calorie constraint.

Chromosome Representation – Each chromosome has 70 genes which takes on binary '0' or '1' indicating if a particular food item is selected. Only 21 genes can be '1'.

Fitness Function – The fitness function is the summation of utility score for the 21 food items represented in each chromosome.

Constraints – There are two constraints. (1) The solution must only select 21 meals (21 ones in the final solution) based on the assumption of 3 meals a day i.e. 21 meals a week. (2) The allocated calorie budget cannot be exceeded. (1) is a hard constraint and (2) is a soft constraint with penalties if violated.

Crossover Operation – The crossover operation uses 2 randomly selected crossover points.

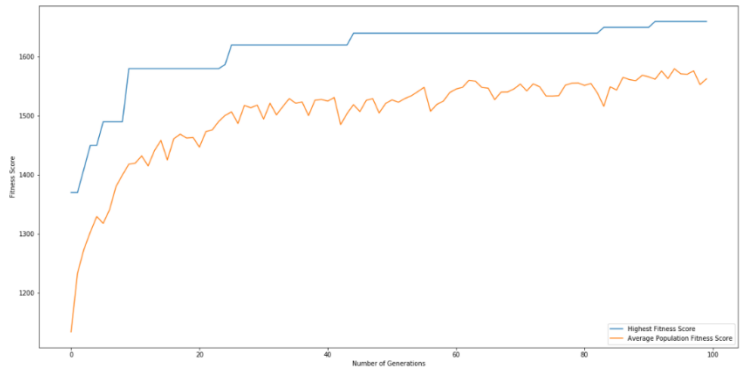
Mutation Operation – The mutation operation random toggles the gene value based on rate of mutation. After mutation, a special handing is implemented to adjust the gene value such that there are only 21 genes with '1'.

Replacement Strategy – The fittest half parent population is combined with the fittest half offspring population to form the next generation.

Parameter and Constraint Settings

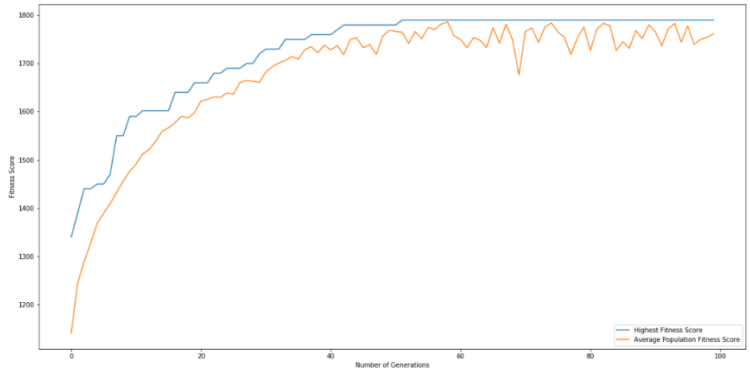
Parameter Description	Initial Run	Final Run
Population Size	50	50
Number of Generations	100	100
Crossover Rate	0.5	0.5
Rate of Mutation	0.1	0.01
Calories (Soft Constraint)	1800	1800

Initial Run Result



A highest fitness score of 1660 is found which is about 20% improvement from the first generation of about 1380. The average daily calories is 1694 which is below the 1800 constraint.

Final Run Result with Rate of Mutation Reduced to 0.01



In the final run, a highest fitness score of 1790 is found with calories within the soft constraint. It is a further 7.8% improvement from the initial run.

4.2 Evolutionary Programming

Similar to Genetic Algorithm, Evolutionary Programming is suitable for this task because the task is a combinatorial optimization problem with a maximum calorie constraint.

Chromosome Representation – Each chromosome has 70 genes which takes on binary '0' or '1' indicating if a particular food item is selected. Only 21 genes can be '1'.

Fitness Function – The fitness function is the summation of utility score for the 21 food items represented in each chromosome.

Constraints – There are two constraints. (1) The solution must only select 21 meals (21 ones in the final solution) based on the assumption of 3 meals a day i.e. 21 meals a week. (2) The allocated calorie budget cannot be exceeded. (1) is a hard constraint and (2) is a soft constraint with penalties if violated.

Mutation Operation – The mutation operation random toggles the gene value based on rate of mutation. After mutation, a special handing is implemented to adjust the gene value such that there are only 21 genes with '1'.

Mutation Rate – The mutation rate is reduced by 10% after every 10% completion of the total generation iterations.

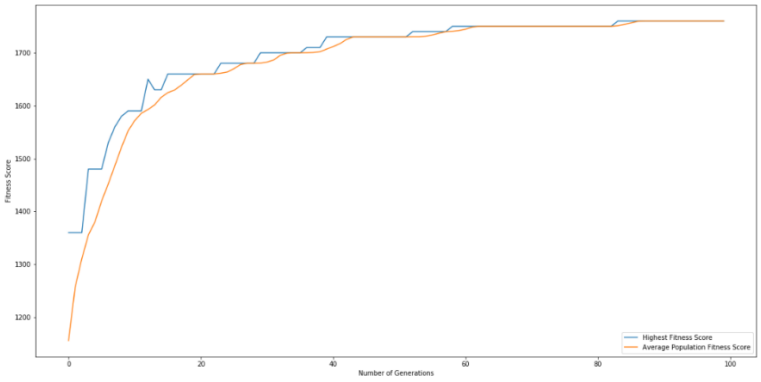
Replacement Strategy – N-tournament is conducted between randomly picked N parents and N offspring and the chromosome with the highest fitness score is selected to form the new generation.

Parameter and Constraint Settings

Parameter Description	Initial Run	Final Run
Population Size	50	50
Number of Generations	100	100
Rate of Mutation	0.1	0.1
Tournament Size	3	10

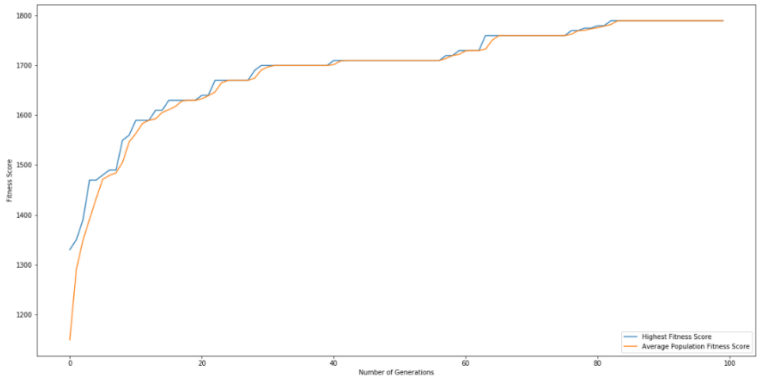
Calories (Soft Constraint)	1800	1800

Initial Run Result



A highest fitness score of 1760 is found which is comparable to the GA result. An interesting observation is that the average population fitness score is closer to the highest score as compared to GA. This could be because the crossover operation in GA adds more randomness to the pool of solution in the generation.

Final Run Result with Tournament Size set to 10



A highest fitness score of 1790 is found which is the same as the GA result. Compared to the initial run where the tournament size was set to 3, the average population fitness score is even closer to the highest fitness score throughout the generations. This however has come at the expense of higher computation cost to compare more chromosomes during the selection for forming a new generation.

4.3 Evolution Strategies

Evolution Strategies Algorithm is suitable for this task because the problem can be defined as finding a fixed-length real-valued vector x that is associated with the optimizing function F(x). The food items are encoded into integer numbers for representation. By mutating and selecting vector x, an optimised solution can be found within constraint.

Chromosome Representation – Each chromosome has 21 genes which represents the combination of 21 food items (3 meals per day × 7 days), each gene is an integer ranging from 0 to 69 which stands for each corresponding food item.

Fitness Function – The fitness function is the summation of utility score for the 21 food items represented in each chromosome.

Constraint – Not to exceeded by 1800 calories × 7 days

Combination Operation – Since the genes takes on discrete values, offspring are generated by selecting the parental values from two parents.

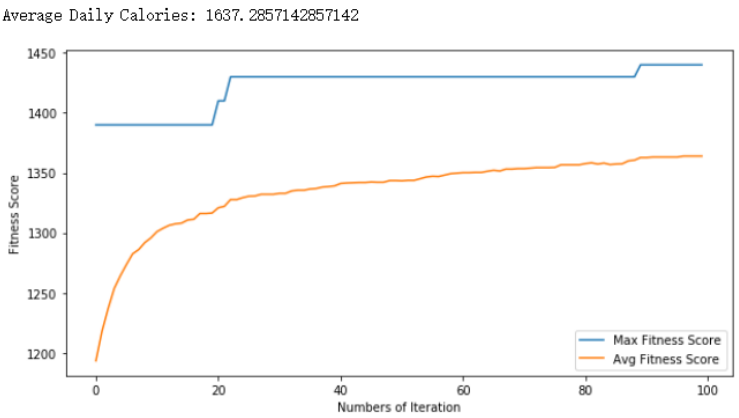
Mutation Operation –The offspring chromosome is created from each parent by adding a random variable from discrete uniform distribution with zero mean and standard deviation to each gene of the chromosomes. The mutation rate is added as an adaptive mutation strategy. The mutation range is kept within the food items range.

Survivor Selection – Both (μ,λ) and (μ+λ) approach were tested.

Parameter Settings

Parameter Description	1st Run	2nd Run	3rd Run	Final Run
Population Size	100	100	100	100
Number of Generations	100	50	50	50
Offspring Size	200	500	200	200
Mutation Operation	add random variables	add random variables	add random variables	add random variables, mutation rate 0.2
Survivor Selection	(μ+λ)	(μ+λ)	(μ,λ)	(μ,λ)

First Run – Best Fitness Score: 1440



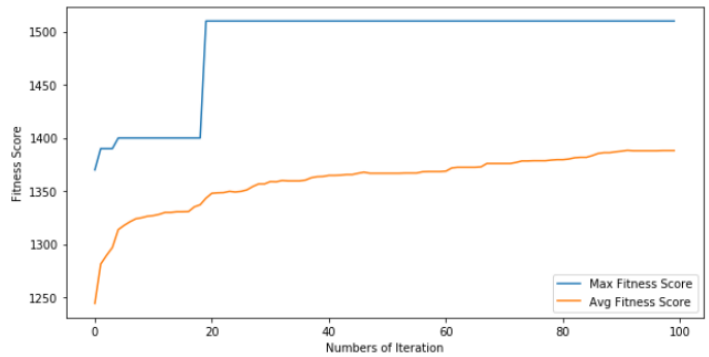
The model Improves slowly, most of the iteration is hitting the local optima and couldn't escape.

Possible Improvements

- Assuming the model doesn't produce enough offspring to find more possible optimal solutions. Producing more offspring will be implemented in the next step.
- Assuming not all the offspring are good enough to iterate to the next generation, the quality of next iteration may be discounted. Selecting less population to reserve 'elites' for next iteration may be a way to solve this assumed problem.

Second Run – Best Fitness Score: 1510

Average Daily Calories: 1700.7142857142858

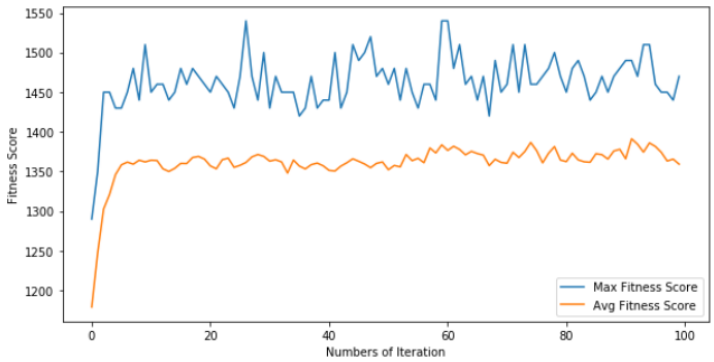


A big lucky move was observed at around 19th iteration. However, after this move, the model is still stuck in local optimum.

Possible Improvement - Try (μ,λ)-selection to add more randomness since it only keeps the mutated offspring to the next iteration instead of including parental chromosomes.

Third Run – Best Fitness Score: 1540

Average Daily Calories: 1639.0

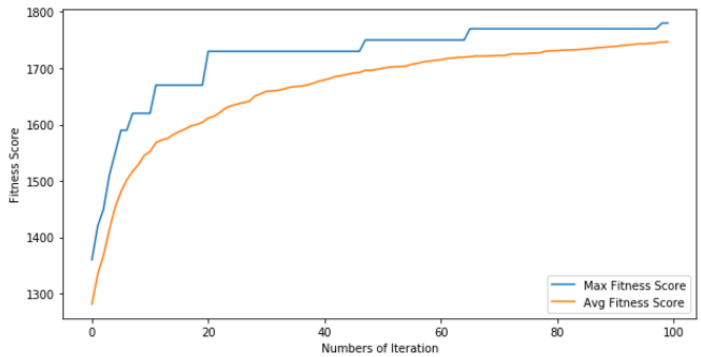


Even though the fitness score has improved by 30 points compared to the previous run, it does not converge and the fitness score is 'unstable'.

Possible Improvement - Add a mutation rate to constrain stochastic moves to some extent.

Final Run – Best Fitness Score: 1780

Average Daily Calories: 1776.142857142857



In the final run, a fitness score of 1780 was achieved which is a 23% increase from the initial fitness score of 1440. Among all the tuning steps, setting the mutation rate to 0.2 yields the highest performance gain.

4.4 Particle Swarm Optimization

PSO is suitable for this task because the task is a combinatorial optimization problem that can be represented as a particle vector, with an objective function to optimize with soft constraints.

In view of the problem statement, the binary variant of PSO developed by Kennedy and Eberhart in 1997 is selected. The specific implementation is supported by the PySwarms python library created by Lester James Miranda.

Modelling – Each particle is a 70-dimension vector representing the 70 types of food that is going to be selected. The value can only be a 0 or 1.

Objective Function – The objective function is the level of utility the user can derive. As the PySwarms library is a minimum optimizer, the level of utility shall be pre-processed to negative integer representation. However, in results table the negative sign is removed since the magnitude is more important for comparison.

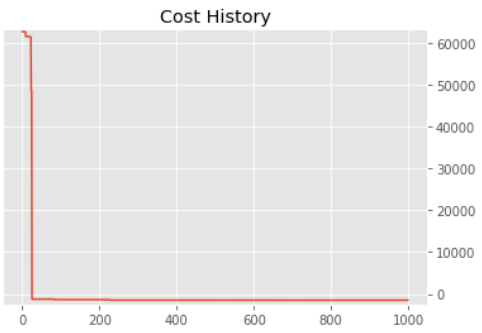
Constraints – There are a couple of constraints modelled. Firstly, the solution must only select 21 meals (21 ones in the final solution) based on the assumption of 3 meals a day i.e. 21 meals a week. Secondly, the allocated calorie budget cannot be exceeded. Both these constraints are soft constraints with respective penalties if violated.

Parameter and Constraint Settings

Parameter Description	Initial Run	Final Run
c1, cognitive parameter	0.5	1
c2, social parameter	0.5	2
w, inertia parameter	0.9	0.9
k, number of neighbors to be considered	30 (entire swarm)	30 (entire swarm)
p, Minkowski p-norm	1 (L1-norm)	1 (L1-norm)
Particle dimension	70	70
Swarm size	30	30
Maximum Calories	1800 (1 day)	1800 (1 day)
Number of iterations	1000	1000

Initial Run

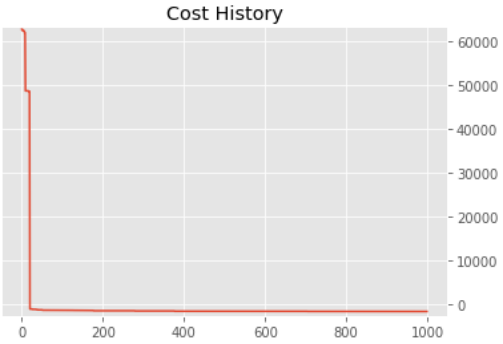
In the initial run, the below cost history and results were obtained. Please note that the PySwarms library is a minimizer and hence the negative sign. Negative sign has been removed in the result table.



Results	Value
Best cost (utility)	1500.0
Average Daily Calories	1780.0

Final Run

After several rounds of tuning, the below cost history and results were obtained. Please note that the PySwarms library is a minimizer and hence the negative sign. Negative sign has been removed in the result table.



Results	Value
Best Utility	1740.0
Average Daily Calories	1779.142857142857

The final model achieved a 16% improvement over the initial model. In both models, it was observed that saturation was achieved early, with relatively little improvement in the remaining iterations. However, as all iterations completed in approximately 4 seconds, no changes were made to the configuration.

4.5 Comparison Between the Four Techniques

All four techniques discussed above produced comparable results. Genetic Algorithm and Evolutionary Programming produces the highest fitness score of 1790. Between the two, Evolutionary Programming is the chosen technique because of its lower computation cost (no crossover operation required) and its ability to produce higher average population fitness score.

Below are the 21 food items recommended by the Evolutionary Programming technique. This list of 21 food items is written to 'selected_food.csv' file which will be used as input for the Daily Meal Planner.

```
Recommended Menu for the week:

Ee Mee Seafood (Fried), Ngoh Hiang Mixed Items, Laksa, Ipoh Horfun, Fried Hong Kong Noodles, Chicken Teriyaki Don, Fried Hokkien Noodles, Braised Duck With Yam Rice, Lor Mee, Fried Carrot Cake, Roti Prata with Egg (2 pieces), Shrimp Fried Rice, Nasi Lemak, Pork Satay with Sauce (10 sticks), Fried Beehoon, Tau Suan, Pizza (2 slices), Carbonara, Butter Crab (500 gram), Mac and Cheese, Fish and Chips

Average Daily Calories: 1787.7142857142858
```

5. Daily Meal Planner

The Daily Meal Planner uses Q-Learning, a form of reinforcement learning to learn the optimized sequence to consume the 21 food items recommended by the Calories and Happiness Optimizer. It avoids sequencing similar food one after another which diminishes the enjoyment for the second food item due to user getting sick of similar food. Reinforcement learning is suitable for this task because user preference changes over time and reinforcement learning can continuously learn the user preference from user feedbacks. Q-learning is selected because we are solving a model free problem and we want the planner to learn an optimal greedy policy.

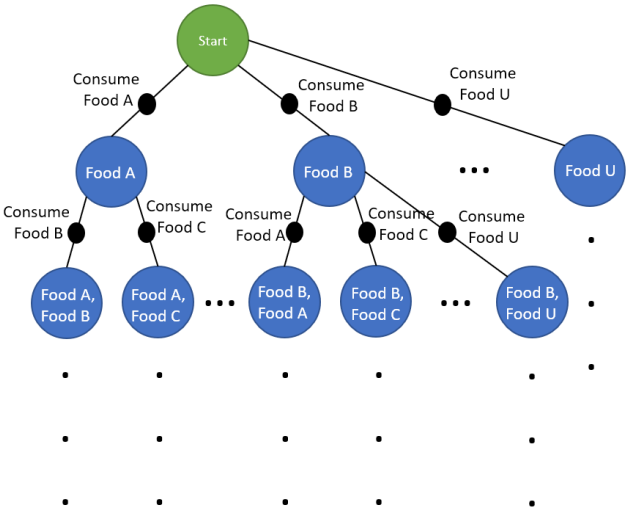
5.1 MDP Modelling

States – States represents the sequence of the food consumption. Examples of states are <Food A, Food B>, <Food A, Food C, Food D> etc.

Action – An action is the action of consuming a specific food item.

State Transition – State transition is deterministic. When an action is performed to consume a specific food, the next state will be all previously consumed food plus the current consumed food.

Reward – Reward is calculated by multiplying user feedback score (after the action of consuming a specific food) with the utility value of the food item. The reward signal is estimated during training.



5.2 Reward Estimation

The main challenge in developing the system is getting real users to give feedbacks after consuming a meal. Even if we can get a large enough pool of users who are willing to help train the system, they only eat 3 main meals a day and it will take a very long time to train the system.

To overcome the problem, the team has come up with a method to estimate the reward signal. The estimation is based on the hypothesis that food with similar main ingredients will have similar carbohydrate, fat and protein composite. The estimated reward signal is the utility score of the food currently consumed multiple by a discount factor determined by the Euclidean distance between the composition of the food item currently consumed and the most recent food consumed in the past.

The approach is to train the system up to a baseline level using the estimated reward signal and later cutover to production to learn the finer user preference.

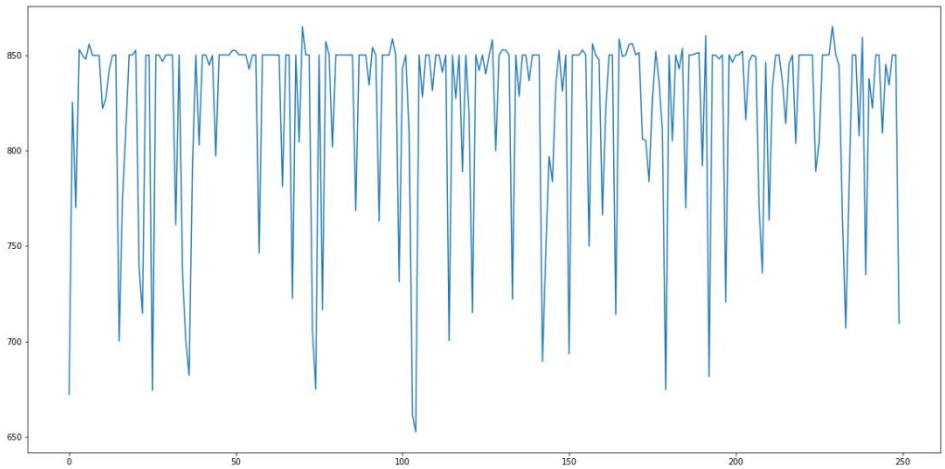
5.3 Training Result

Parameters

eps	0.05
gamma	0.95
Episodes	250,000
Initial Learning Rate	1.0
Minimum Learning Rate	0.003

Results

The total rewards score starts from 672 points in the first iteration and quickly climb to the region of 850 points. The rewards score largely hovers around 850 points with occasional dips as random moves were being explored.



Below is an output of the Daily Meal Planner.

```
Day 1 Eating Plan
Fried Hokkien Noodles, Braised Duck With Yam Rice, Ee Mee Seafood (Fried)

Day 2 Eating Plan
Ngoh Hiang Mixed Items, Fried Hong Kong Noodles, Laksa

Day 3 Eating Plan
Ipoh Horfun, Chicken Teriyaki Don, Fish and Chips

Day 4 Eating Plan
Lor Mee Fried Carrot Cake, Roti Prata, with Egg (2 pieces)

Day 5 Eating Plan
Shrimp Fried Rice, Nasi Lemak, Pork Satay with Sauce (10 sticks)

Day 6 Eating Plan
Fried Beehoon, Tau Suan, Pizza (2 slices)

Day 7 Eating Plan
Carbonara, Butter Crab (500 gram), Mac and Cheese
```

6. Daily Exercise Planner

In addition to food sequence recommendation and calorie tracking, there might be other goals that the users want to achieve in addition to leading a healthy lifestyle. One typical goal is weight loss, especially after a fair number of sumptuous meals with family and friends.

The typical approach to this would be to estimate a person's total daily energy expenditure using a calculator that can be commonly found in various websites dedicated to health and fitness. These calculators typically relied on the basal metabolic rate estimation formula like the Harris-Benedict equation, the Mifflin St Jeor equation or the Katch-McArdle formula and request users to input information like sex, age, weight, height and level of activity. While this formula provides a fair estimation, some downsides include:

- This is a one-off calculation, no historical data is considered
- Level of activity can mean different things to different people
- It does not consider other factors like lifestyle and metabolic rate

The team then set out to explore solutions to provide a better estimate of behaviours and activities that can lead to better weight loss outcomes. The team propose to use a neural network model to model the degree of weight loss possible given past calorie intake and exercise activities and then use an evolutionary algorithm-based optimizer to plan exercise activities. For this task, the multilayer perceptron and differential evolution algorithm is selected for concept implementation and will be further elaborated in subsequent sections.

Due to the difficulty of collecting calorie intake and exercise activity information, the team decided to base the concept model on the dataset provided by ChrisBow on Kaggle as mentioned in earlier sections to prove the solution concept.

6.1 Data Pre-processing

The following steps are done in data pre-processing. These functions are wrapped inside a function.

1. Identify and remove non-features including ['Stone', 'Pounds', 'Ounces', 'weight_oz', 'cals_per_oz', 'five_donuts', 'wine', 'prot', 'weight', 'change', 'Date'] for the input data
2. Set 'change' as the target variable
3. Perform data split into 80% training and 20% test set with no shuffle. This is because the dataset is a type of longitudinal data
4. Standardization is fitted to the training set and applied to both training and test set

6.2 Multilayer Perceptron Model and Particle Swarm Optimization Setup

The following steps are performed during the setup of the multilayer perceptron and each step corresponds to a function definition.

1. Calculate the number of parameters in the multilayer perceptron including number of input and bias nodes in the input, hidden and output layers
2. Perform forward propagation using PSO as optimizer
3. Evaluate particle loss function (using Mean Squared Error, MSE) on training set
4. Perform prediction on test set and evaluate test set MSE

Typically, multilayer perceptrons are trained using forward propagation and backpropagation. In this assignment, the team demonstrates that particle swarm optimization can also be used as a method to tune the node weights and optimize the loss function. One advantage is that PSO is easy to implement and do not require functions to be differentiable as compared to backpropagation. Moreover, it is relatively easy to parallelize PSO operations.

The below table shows the Neural Network parameters.

Parameters	**Value**
Number of input nodes	3
Number of hidden layers and nodes	1, 20
Number of output nodes	1
Hidden layer activation function	Leaky ReLU

The PSO Model is as below.

Modelling – Each particle is a 101-dimension vector representing the 101 neural network parameters.

Objective Function – The objective function is to minimize the mean squared error between the training set target and predicted output target.

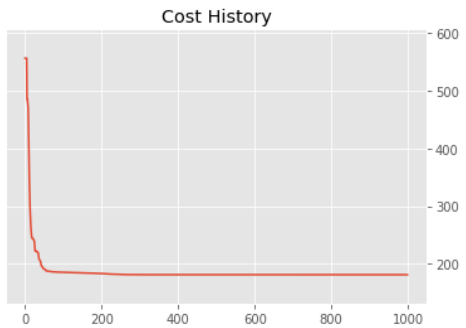
Constraints – There are no particular constraints added.

6.3 Model Training

Initial Model - In the initial model, the below parameters were chosen.

Parameters and Constraints	**Value**
c1, cognitive parameter	0.9
c2, social parameter	0.1
w, inertia parameter	0.9
Particle dimension	101
Swarm size	100
Number of iterations	1000

The below cost history and results were obtained.



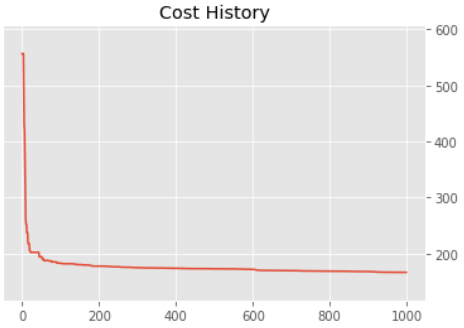
Results	**Value**
Best cost (Training MSE)	181
Validation MSE	271.6631037246394

Overfitted Model – It is of interest to note that during the tuning of the model with PSO, the challenge of overfitting is there as well.

The overfitted model can be observed with the below parameters.

Parameters and Constraints	Value
c1, cognitive parameter	0.1
c2, social parameter	0.9
w, inertia parameter	0.9
Particle dimension	101
Swarm size	100
Number of iterations	1000

The below cost history and results were obtained.



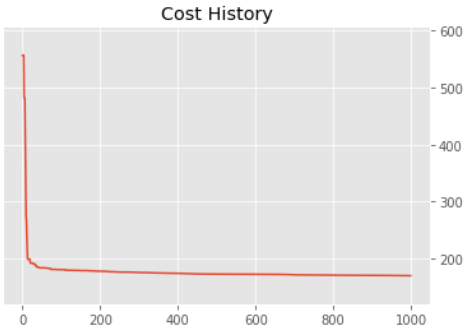
Results	Value
Best cost (Training MSE)	167
Validation MSE	542.7309392335013

Despite a low cost of 167, which as it turns out is the lowest value observed, the test set MSE is much higher than that observed in the initial model. It goes to show that the best cost may not be as desirable when we use metaheuristics like PSO to train neural networks as it may not make the model generalizable and more experiments have to be run.

Final Run – After several rounds of tuning, the best parameters to get the lowest MSE are as below.

Parameters and Constraints	Value
c1, cognitive parameter	0.5
c2, social parameter	0.9
w, inertia parameter	0.9
Particle dimension	101
Swarm size	100
Number of iterations	1000

The below cost history and results were obtained.



Results	Value
Best cost (Training MSE)	170
Validation MSE	229.73113330078814

Comparison with baseline Linear Regression – The team built a baseline Linear Regression model to understand whether the tuned model is effective or not.

Linear Regression Baseline MSE	Final Multilayer Perceptron MSE
279.38889160496416	229.73113330078814

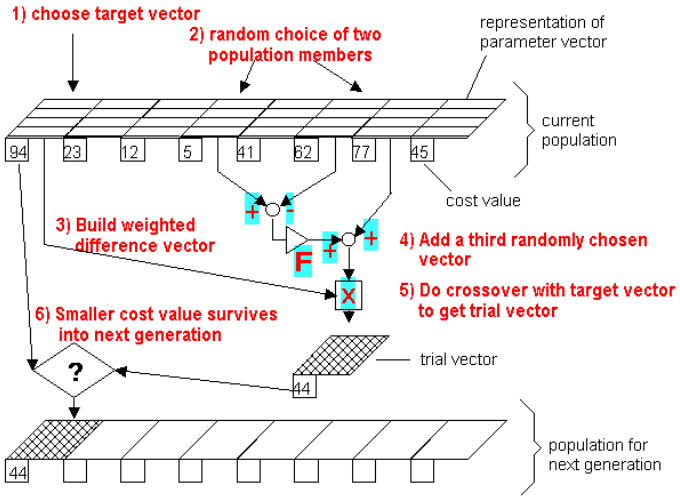
It was observed that the Final Multilayer Perceptron model showed a reduction of 17.77% in MSE over the Linear Regression Baseline and can be determined to be a better model by MSE.

6.4 Workout Optimization with Differential Evolution

Developed by Price and Storn, Differential Evolution is a population-based evolutionary algorithm. Being a metaheuristic, the algorithm is suitable for determining the best set of exercises to achieve the target weight loss given a known calorie budget in this assignment.

The figure below from the documentation of pygmo, a python library, provides additional details of the algorithm. Compared to Genetic Algorithm, the major differences are in step 3 and 4, where the algorithm computes a weighted difference vector from the parents and adds a randomized vector before performing crossover.

The implementation used in this assignment is provided by the Scipy library.



Source: https://esa.github.io/pagmo2/docs/cpp/algorithms/de.html

Modelling – Each parameter vector is modelled as the combination of calorie intake and the types of exercises. In this case there are two exercises, making it a total vector size of 3.

Objective Function – The objective function is to minimize the difference between target daily average weight loss and predicted weight loss based on historical data recorded. It has been configured to cater to the varied weight loss goals as some people would like to set more aggressive goals while some prefer more gradual weight loss.

A confidence factor has also been applied in the loss function. The confidence factor represents the uncertainty that the model is unable to capture, such as data entry errors, user estimation errors and other uncaptured variables. In production, this confidence factor can be tweaked by the system depending on the feedback from the user (accurate or not accurate).

Constraints – There are a couple of constraints being modelled. Firstly, are the bounds for calorie intake and the exercises. In particular, due to differential evolution accepting real-values, the exercise binary variables (such as run and brisk walk) need to be rounded to the nearest integer (0 or 1). Secondly, the calorie variable is known beforehand and pegged to the known value and needs to be standardized before use in the optimizer.

The following parameters are used.

Parameters and Bounds	**Value**
Population Size	15 (default)
Strategy	best1bin (default)
Initialization	latinhypercube (default)
Max Iterations	1000 (default)
Bounds: Calorie	(calorie,calorie+1)
Bounds: (brisk) walk	(0,1)
Bounds: run	(0,1)

Rather than looking at the objective function optimization history, it is better to understand how the optimizer works in some simulated scenarios to observe the effectiveness of the optimizer.

Scenario 1: Sample Menu Recommendation

In this scenario, a user wants to achieve a daily average weight loss target of 9 ounces (approx. 0.25 Kg) based on the simulated weekly menu created using the PSO variant of the calorie recommender based on a daily average calorie budget of 1800 calories and the diet data (calorie intake, weight change in ounces, if the user went for at least one brisk walk of over 20 minutes and if the user went for 1 run of 2.5 miles) from 30-July-18 to 19-Dec-18.

	recommended_food	calories	recommended_exercise
0	Ipoh Horfun | Ee Mee Seafood (Fried) | Butter ...	2135	walk
1	Braised Duck With Yam Rice | Lor Mee | Chicken...	1579	
2	Pork Satay with Sauce (10 sticks) | Fried Hokk...	1877	
3	Laksa | Tau Suan | Pork Porridge	1257	
4	Ngoh Hiang Mixed Items | Alfredo Pasta | Roti ...	1951	walk
5	Carbonara | Roti Prata Plain (2 pieces) | Mac ...	1580	
6	Mee Rebus | Nasi Lemak | Fish and Chips	2075	walk

Using a 0-based index in the table above, we can see that the Workout Planner recommended the user to go for brisk walks on Day 0, Day 4 and Day 6 in addition to adhering to the sample menu.

Scenario 2: Festive Season Simulation

During the festive seasons, there's typically more sumptuous meals with family and friends that might provide greater challenge to weight loss goals. Hence, we simulate this scenario below to observe the recommendations of the workout planner. Say the user went for 3 days of festive meals with the corresponding changes in the menu:

- Day 1: calorie intake increased from 1579 calories to an estimated 2500 calories
- Day 2: calorie intake increased from 1877 calories to an estimated 2700 calories
- Day 4: calorie intake increased from 1951 calories to an estimated 2900 calories

	old_calories	old_recommended_exercise	new_calories	new_recommended_exercise
0	2135	walk	2135	walk
1	1579		2500	walk | run
2	1877		2700	run
3	1257		1257	
4	1951	walk	2900	run
5	1580		1580	
6	2075	walk	2075	walk

In the table above, we can see that on the expected days of festive meals (Days 1, 2 and 4), the workout planner recommended more brisk walks and runs. In particular on Day 4, increased the exercise intensity from brisk walk to run. This is an acceptable outcome and demonstrates that the planner is adapting as expected to changes in calorie intake.

6.5 Simulation of Workout Planning Optimization with New Activity Features

To further demonstrate how the PSO-evolved multilayer perceptron weight loss regressor and the differential evolution workout optimizer works, the team generated new data to simulate the introduction of a new exercise by the user - swimming.

Suppose the user decided to start swimming on 20-Dec-18, with the following simulation settings for 20 days (till 8-Jan-19) for a comparison with the previous 20-day time period (30-Nov-18 to 19-Dec-19).

- User swam (20-laps) for 9 times in the 20-day period between 20-Dec-18 and 8-Jan-19
- User decreased average intake of calories during this time but still did not follow the recommended 1800 daily average calorie intake meals
- User experienced weight loss but still does not achieve the average weight loss target
- User decided to start following the 1800 calorie recommendation in the previous scenarios
- System has an additional 20 days of records for 'weight_oz', 'calories', 'walk', 'run', 'swim' and 'change'. All other columns that are not used are represented with '-9999'. For previous data records of 'swim' the value of '0' is imputed by the system since it is as good as the user not swimming in the past

The below two tables summarizes the configuration of the simulation.

Variable/Time Period	**30-Nov-18 to 19-Dec-19**	**20-Dec-18 to 8-Jan-19**
Average Calorie Intake	3000	2417
Average Weight	2672 Ounces, 75.7 Kg	2570 Ounces, 72.9 Kg

Results after swimming 9 times in 20 days	**Value**
Target Average Weight Loss in 20 days	5 Kg

Actual Average Weight Loss in 20 days	2.8 Kg

Neural Network Adaptation

During coding, the team has taken care to write functions that can be easily reused in the situation with no modifications where new features are introduced. In this assignment, the functions for data pre-processing, neural network parameter calculations, PSO training and prediction and differential evolution optimization will remain unchanged. The only modification would be PSO hyperparameter tuning that can be implemented as a new feature in production in the future.

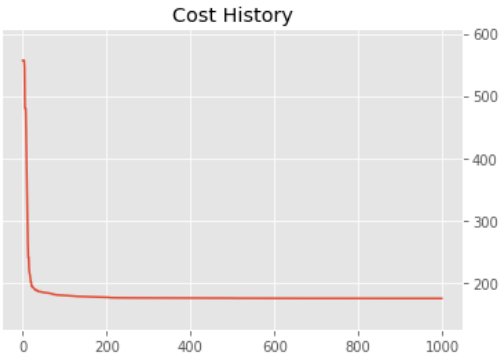
With the introduction of a new feature 'swim', the multilayer perceptron parameters will be as below.

Parameters	**Value**
Number of input nodes	4
Number of hidden layers and nodes	1, 20
Number of output nodes	1
Hidden layer activation function	Leaky ReLU

Model Retraining – During training, the hyperparameters are as follows.

Parameters and Constraints	**Value**
c1, cognitive parameter	0.5
c2, social parameter	0.5
w, inertia parameter	0.9
Particle dimension	121
Swarm size	100
Number of iterations	1000

The below cost history and results are obtained.



Results	**Value**
Best cost (Training MSE)	175
Validation MSE	319.055471663071

Scenario 1: Sample Menu Recommendation Rerun

Based on the newly updated weight loss regressor model, the differential evolution workout planner optimizer determined that in addition to following the walks previously recommended on days 0, 4 and 6, additional swims are required to reach the target daily average weight loss.

	recommended_food	calories	recommended_exercise
0	Ipoh Horfun | Ee Mee Seafood (Fried) | Butter ...	2135	walk
1	Braised Duck With Yam Rice | Lor Mee | Chicken...	1579	
2	Pork Satay with Sauce (10 sticks) | Fried Hokk...	1877	swim
3	Laksa | Tau Suan | Pork Porridge	1257	
4	Ngoh Hiang Mixed Items | Alfredo Pasta | Roti ...	1951	walk | swim
5	Carbonara | Roti Prata Plain (2 pieces) | Mac ...	1580	
6	Mee Rebus | Nasi Lemak | Fish and Chips	2075	walk

This shows that the differential evolution workout optimizer, in tandem with the updated PSO-evolved regressor model, has modified its recommendation according to insights learned from the additional data and features, which is the expected outcome.

Scenario 2: Festive Season Simulation Rerun

Re-running the Festive Season Simulation again, we can see that the differential evolution workout optimizer has recommended additional exercises or increased the intensity of exercise on days where the user is expected to have sumptuous meals (i.e. days 1, 2 and 4) based on the insights learned from the additional data and features.

	old_calories	old_recommended_exercise	new_calories	new_recommended_exercise
0	2135	walk	2135	walk
1	1579		2500	run
2	1877	swim	2700	run | swim
3	1257		1257	
4	1951	walk | swim	2900	run | swim
5	1580		1580	
6	2075	walk	2075	walk

This scenario shows that the differential evolution workout optimizer is working as expected.

Before closing the section, the team would like to highlight that **the models presented should be viewed as an assistive tool intended to assist users derive deeper insights and is not a substitute for professional medical consultation, diagnosis or treatment.**

7. Understanding and Findings

Genetic Algorithm and Evolutionary Programming performs very well in solving combinatorial optimization problem. The heuristic plus random search is an efficient way to perform searching in a large solution space. It was observed that for Calories and Happiness Optimization task, crossover operation in Genetic Algorithm did not really help.

Q-learning can be used for problems with large number of states with a minor adaption in the implementation of the Q-table. The Daily Meal Planner has more than 5.1E+19 number of states for the different possible sequences of 21 food items. Instead of initiating a super large Q-table with all zeros, a dictionary object (in python) is initialised on the fly when a particular state-action value is to be updated.

Evolution Strategies could coevolve the mutation step size along with the solution. However, the implemented model uses discrete value to represent food items, the gradual convergence of changing a certain discrete value does not contribute much to adapt higher utility score. One possible adaptive method might be re-arranging the food item distribution. The mean of the discrete standard

distribution is the food with the highest utility score. Food with lower utility scores is located further from the mean. The mutation step size could thus control the corresponding utility score to best fit the fitness function.

Particle Swarm Optimization is a very valuable tool for use in optimization and can be used for both binary or real-value vector variables. It is however noted that it tends to converge to local minima quickly (as shown in the rapid drop in cost histories across various models presented) and several rounds of hyperparameter tuning are recommended to get optimal results.

Particle Swarm Optimization has also been demonstrated to be an alternative to backpropagation when training neural networks and can be considered depending on requirements. However, the challenge of overfitting is always present when it comes to model training in machine learning.

Differential Evolution has been demonstrated to be a useful and fast optimizer as seen in the workout optimization.

Last but not least, it has been demonstrated that evolutionary algorithms are adaptable and help models to learn and adapt to new data and features.

8. Conclusion

The team has successfully implemented an Intelligent Self-Learning System which recommends to the user a highly personalised daily eating and exercise plan. The system was implemented using various reinforcement learning and evolutionary learning techniques, namely: Q-Learning, Genetic Algorithm, Evolutionary Programming, Evolution Strategies, Particle Swarm Optimization and Differential Evolution. When live data from IoT Sensors and user feedback is available and is integrated in a production system, the system will be capable of continuous self-learning and self-evolving according to the changing environment such as changing user food preference, new food items and new exercises.

References

ChrisBow (2019). 2018 calorie, exercise and weight changes, Kaggle, https://www.kaggle.com/chrisbow/2018-calorie-exercise-and-weight-changes

J. Kennedy and R.C. Eberhart, “A discrete binary version of particle swarm algorithm,” Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1997

Miranda, (2018). PySwarms: a research toolkit for Particle Swarm Optimization in Python. Journal of Open Source Software, 3(21), 433, https://doi.org/10.21105/joss.00433

Scipy (2019) SciPy Reference Guide, Optimization and Root Finding, Differential Evolution, https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

Storn, R and Price, K, Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 1997, 11, 341 - 359.

Wikipedia (2019), Basal metabolic rate, https://en.wikipedia.org/wiki/Basal_metabolic_rate

Annex A – Team Members Individual Reflection

CA2 - Individual Project Report

Your Name:	Alfred Tay Wenjie
Certificate:	Graduate Certificate in Self Learning System

1. Personal contribution to the group project

Ideate and formalize problem statement. Perform overall high-level solution design. Model and implement Genetic Algorithm and Evolutionary Programming (develop custom code using python) for Calories and Happiness Optimizer. Model and implement Q-Learning (develop custom code using python) for Daily Meal Planner. Testing and parameter tuning. Compile food listing dataset. Report writing on the above portions. Facilitated project discussions, scoping of work and tracking of project progress.

2. What you have learnt from the project

I have learnt that Genetic Algorithm and Evolutionary Programming is very useful for solving combinatorial problems. How the solution is represented as chromosome will affect the performance of the algorithm. Genetic Algorithm and Evolutionary Programming combines heuristic search and random search which is similar to how human being solves problem. We use our experience to determine a general approach (heuristic) and look for good solution by experimenting (random search) with a few possible solutions.

Reinforcement learning continuously interact with the environment to learn the action which provides the highest current and future rewards. Unlike supervised learning and unsupervised learning, reinforcement learning can adapt to changing environment and has many potential applications. The challenges are in modelling the action and state transition, designing the reward function and getting frequent real-world reward signal.

3. How you can apply the knowledge and skills in other situations

One potential application of reinforcement learning is in organisation learning. Many of the organisation's knowledge exist in the form of tacit knowledge residing in its directors, managers and rank and file employee. As people come and go, these knowledges are lost. Also, traditional knowledge management techniques such as keeping documentations is passive and difficult to search. The key to implementing RL in an organisation is having a digital twin of the organisation's operating environment. After that, selected business processes can be modelled into MDP and RL agents can be trained using simulated data in the digital twin virtual environment.

CA2 - Individual Project Report

Your Name:	Wang Zilong
Certificate:	Graduate Certificate in Self Learning System

1. Personal contribution to the group project

Designed the Calories & Happiness Optimizer with Evolution Strategies, tried different method to implement the algorithm. Writing the report to illustrate design and tuning steps. Code testing in Windows and Linux.

2. What you have learnt from the project

For the Calories & Happiness Optimizer section, I have learned that Evolutionary Computation is a good area to solve these kinds of optimization problems. I have learned different algorithms and how to implement them in the process, such as Genetic Algorithms, Evolutionary Programming, Evolution Strategies, and Particle Swarm Optimization. By coding Evolution Strategies from scratch, I have a deeper understanding of the algorithms. By comparing with other algorithms, I understood each of their advantage and limitation, thus could know how to implement them in the future potential business scenario. For the Daily Meal Planner and Daily Exercise Planner sections that I learned from my peers, I understood that the model-free Q-learning is flexible for optimization problems, and how to use Particle Swarm Optimization to tune a neural network.

3. How you can apply the knowledge and skills in other situations

In the banking industry, employee scheduling problems in call centers and clearing centers could be solved using Evolutionary Computation. Scheduling problems involves so many constraints, different employees have different operation authorities and different individual rest time. the combinations of employees matter, in terms of collaboration and fulfilment. Rule-based scheduling system is limited in finding the optimum and can only be semiautomatic. With the help of other machine learning algorithms, Evolutionary Computation could generate an optimized scheduling table as long as constraints are well defined. The fitness function is finding the lowest human resource cost within a certain workload output. Besides Evolutionary Computation, can apply reinforcement learning such as Q-learning to learn a series of transaction behavior and find the best strategy.

CA2 - Individual Project Report

Your Name:	Wong Yoke Keong
Certificate:	Graduate Certificate in Self Learning System

1. Personal contribution to the group project

I have contributed to the following activities:

- Project brainstorming and research
- Modelling, implementation and testing of Particle Swarm Optimization for Food Recommendation
- Data cleaning and transformation for diet dataset and establishing baseline linear regression model for weight loss prediction
- Modelling, implementation, testing and benchmarking of Multilayer Perceptron Regression trained using Particle Swarm Optimization for weight loss prediction
- Modelling, implementation and testing of workout planning optimization using Differential Evolution
- Planning and implementation of simulation scenarios for workout planning optimization to evaluate the optimizer in realistic situations
- Project writing for the respective report sections
- Testing and packaging of the notebook scripts, environment setup and validation

2. What you have learnt from the project

During the project, I gained a deeper insight into the applications of evolutionary computation algorithms in areas like combinatorial, numerical optimization and machine learning.

In terms of modelling, I have learnt to decompose and represent business problems into the solution domain (like chromosomes and particles), objective function for evaluation (like fitness and loss functions) and constraints.

In terms of implementation, I have a deeper understanding about the strengths and weaknesses of different evolutionary algorithms and handle challenges like local optima and overfitting.

In terms of scenario planning, I gained a better appreciation on the use of simulation to understand how to evaluate optimizers under different business scenarios faced by the user.

3. How you can apply the knowledge and skills in other situations

I will be able to use the knowledge gained to solve combinatorial optimization problems like scheduling, assignment and routing problems using evolutionary techniques learnt. I will also be able to use the knowledge gained to solve numerical optimization challenges in conjunction with current machine learning techniques for example, optimizing cost functions in neural networks.