MIS780 – Advanced AI for Business

Week 8

Evolutionary Computation: Genetic Algorithms

Dr Anagi Gamachchi

Discipline of Information Systems and Business Analytics, Deakin Business School

DEAKIN BUSINESS SCHOOL

AACSB ACCREDITED

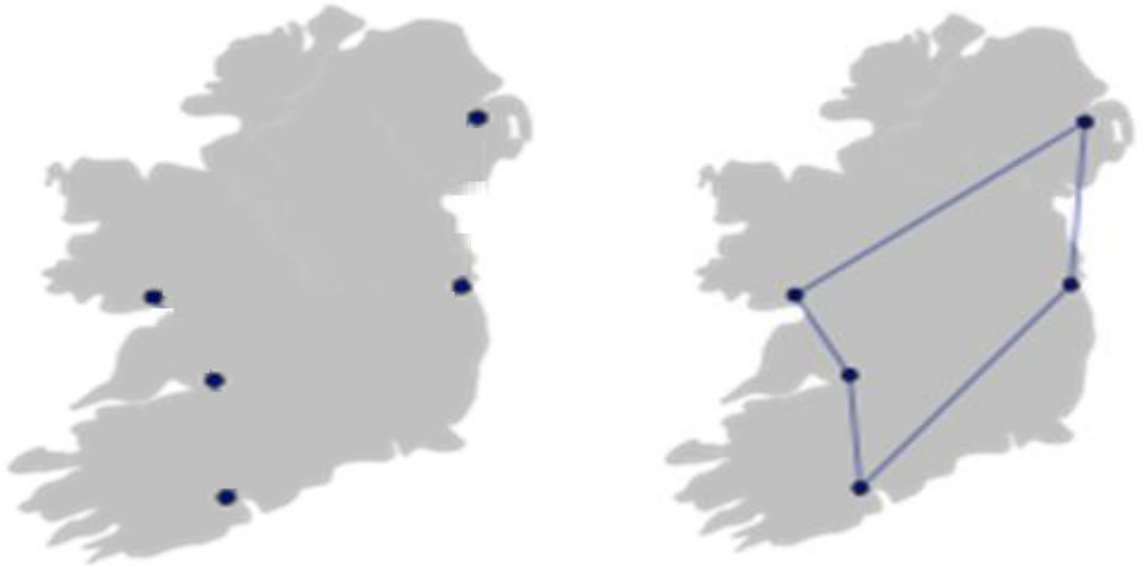
EFMD EQUIS ACCREDITED

Outline

- Travel Salesman Problem
- Machine Learning Optimization

Traveling Sale Man Problem

- The goal of the TSP is to find the most economical way to travel through a select number of "cities" with the following restrictions:
  - must visit each city once and only once
  - must return to the original starting point

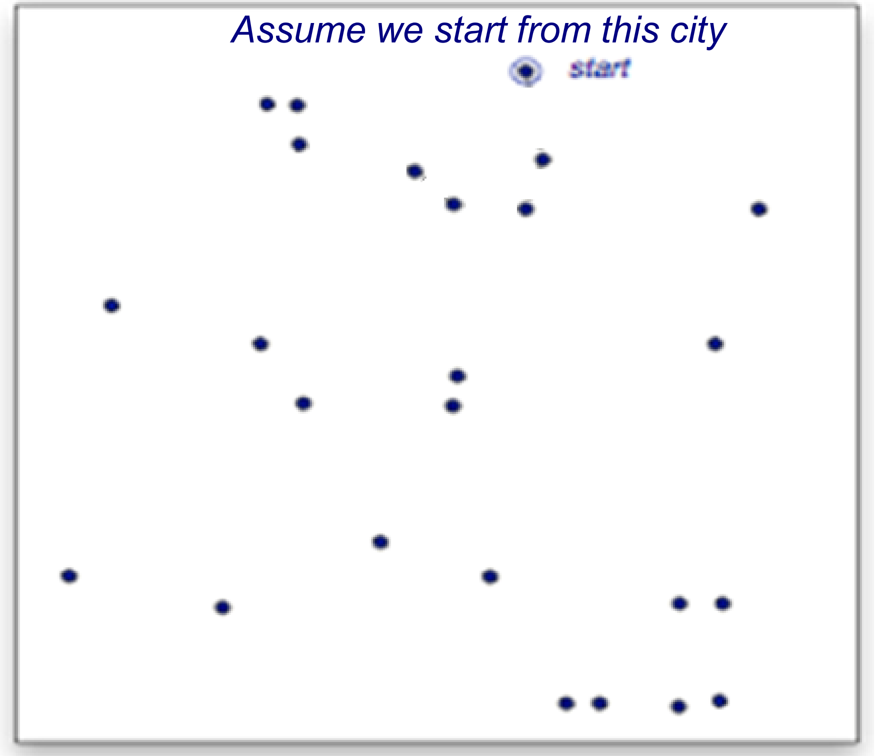


What would be the optimal travel path?

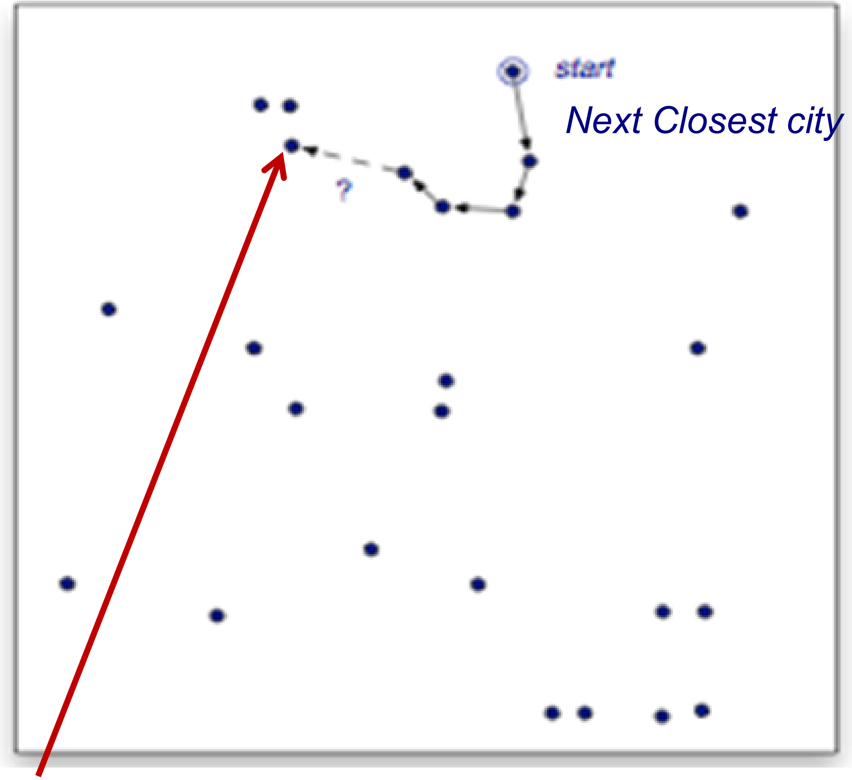
Traveling Sale Man Problem (cont.)

- As we add more cities to our tour, it is much harder to figure out the optimal tour:

What would be the optimal travel path?



25 cities



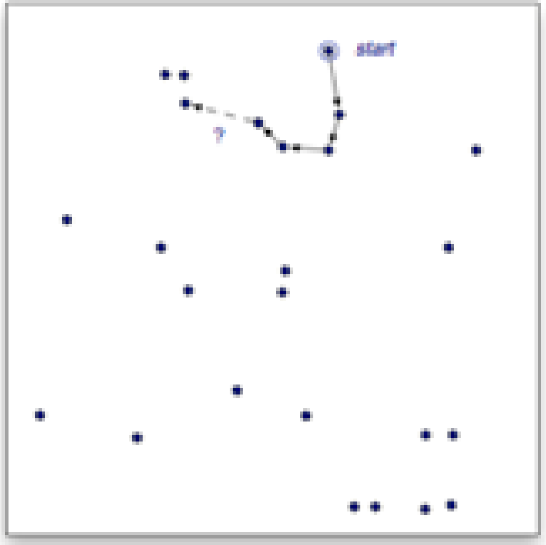
Hint: going to this city does not lead to the shortest tour...

TSP Overview (cont.)

Can we find the optimal path by trying all possible paths?

- The **number of possible path grows incredibly quickly** as we add cities to the map

#cities	#tours
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440



How many possible path for 25 cities?

The number of tours for 25 cities:

310,224,200,866,619,719,680,000

How about 100 of cities?

How can we find the optimal path effectively?

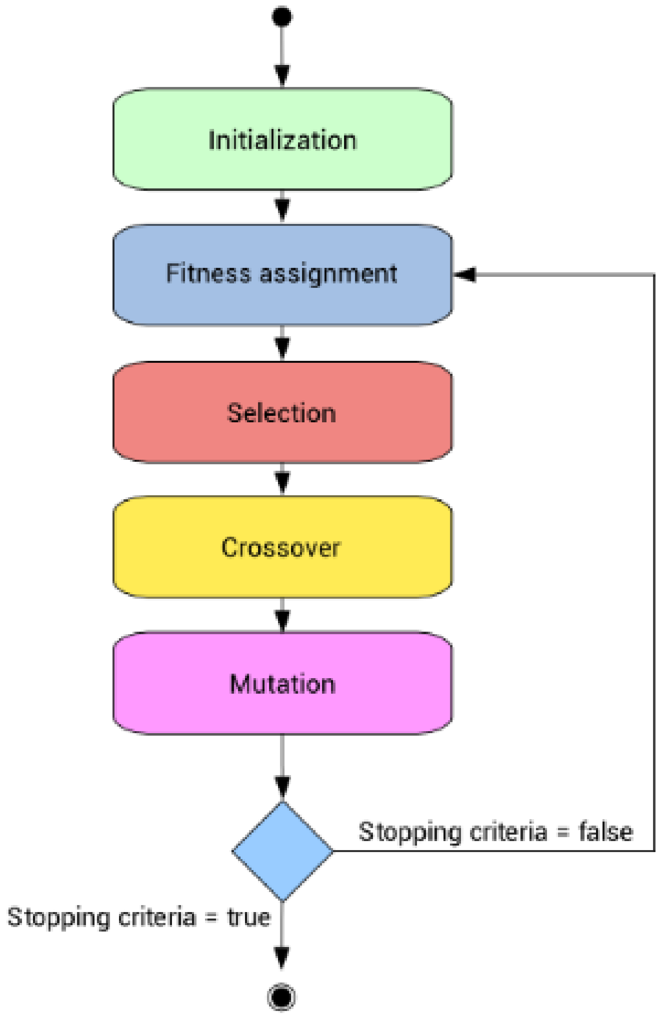
Explorations in Computing: An Introduction to Computer Science, by John S. Conery

Genetic Algorithm

- **Genetic algorithms** are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as ***inheritance***, ***mutation***, ***selection***, and ***crossover***.
  - Developed by John Holland, University of Michigan (1970's)
  - Provide efficient, effective techniques for optimization and machine learning applications
  - Widely-used today in *business*, *scientific* and *engineering* applications.

"Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime."

- Salvatore Mangano
Computer Design, May 1995



Genetic Algorithm - Initialization

- Assume that we have eight cities as below:

1) London	3) Dunedin	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

Problem: find a shortest path to all cities, which goes through each city only once.

- **Step 1**: **Initialization**
  - Representation is an ordered list of city numbers (randomly generated)

Population of size 4 (possible paths)

CityList1	(3 5 7 2 1 6 4 8)
CityList2	(2 5 7 6 8 1 3 4)
CityList3	(4 2 1 5 7 8 3 6)
CityList4	(6 5 1 2 8 4 7 3)

CityList1 → **Chromosome** (one possible solution) of size 8 (number of cities)

Wendy Williams. Genetic Algorithms: A Tutorial

Genetic Algorithm – Fitness Assignment

- **Step 2**: **Compute fitness** values – total distances travel through all cities.

  CityList1 (3 5 7 2 1 6 4 8) - Distance: 33,132 km

  CityList2 (2 5 7 6 8 1 3 4) - Distance : 23,242 km

  CityList3 (4 2 1 5 7 8 3 6) - Distance : 42,143 km

  CityList4 (6 5 1 2 8 4 7 3) - Distance : 16,843 km

  Assume we know the distance between cities

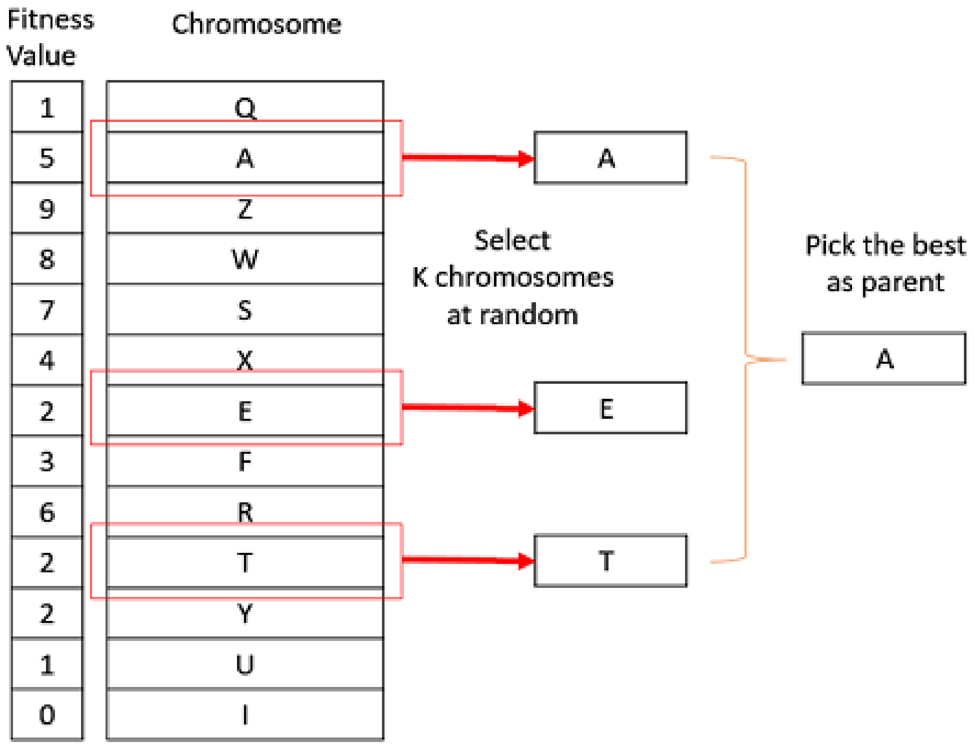
  – Since the objective is to minimize total distance, the **fitness** is defined as

  Fitness = -1*Distance

  (The higher the fitness value, the better path)

Genetic Algorithm – Selection

- **Selection** is a procedure of picking parent chromosomes to produce off-spring for the next generation.
  - The idea is to select the fittest individuals and let them pass their genes to the next generation
  - Two of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.



https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm

Genetic Algorithm – Selection (cont.)

- **Step 3**: **Select parents** to produce the next populations.

CityList1 (3 5 7 2 1 6 4 8) - fitness: - 33132

CityList2 (2 5 7 6 8 1 3 4) - fitness : -23242

CityList3 (4 2 1 5 7 8 3 6) - fitness : -42143

Select using **roulette wheel** or **tournament** approach

CityList4 (6 5 1 2 8 4 7 3) - fitness : -16843

Directly Selected (*to be added directly to the new population later*) - **elitism**

CityList1 -> Parent1 (3 5 7 2 1 6 4 8)

CityList2 -> Parent2 (2 5 7 6 8 1 3 4)

Selected individuals as parents.

Every parents produce one child solution.

Genetic Algorithm – Crossover

- **Step 4: Crossover** process is a permutation of the list of cities.

Parent1 (3 5 7 2 | 1 6 4 8)

Parent2 (2 5 7 6 | 8 1 3 4)

→

Is this a valid solution?

Child (2 5 7 6 1 6 4 8)

These don't go through all the cities **and** they visit some cities twice, violating multiple conditions of the problem.

  - **Partially Mapped Crossover:** randomly picks one crossover point, but unlike one-point crossover it doesn't just swap elements from two parents, but instead swaps the elements within them

P1 (3 5 7 2 | 1 6 4 8)
P2 (2 5 7 6 | 8 1 3 4)

P1 (2 5 7 3 | 1 6 4 8)
P2 (2 5 7 6 | 8 1 3 4)

P1 (2 5 7 3 | 1 6 4 8)
P2 (2 5 7 6 | 8 1 3 4)

→ Child (2 5 7 6 1 3 4 8)

https://stackabuse.com/traveling-salesman-problem-with-genetic-algorithms-in-java/

Solving TSP using GA

- **Step 5**: **Mutation** (randomly) **Swap Mutation**

(for permutation based encoding)

Child (2 5 7 6 1 3 4 8) → Child (2 4 7 6 3 1 5 8)

– A low mutation rate (e.g. 0.01) is usually used to avoid changing too much the gene of good individuals.

Next, GO BACK TO STEP 2 (fitness assignment, etc.) with the new population until stopping criteria (Num. of Iteration) is met.

Individual with **highest fitness in the final population** is returned as **final solution**

Next Generation Population

Directly Selected Previously → CityList4 -> NewList1 (6 5 1 2 8 4 7 3)

Child1 -> NewList2 (2 4 7 6 3 1 5 8)

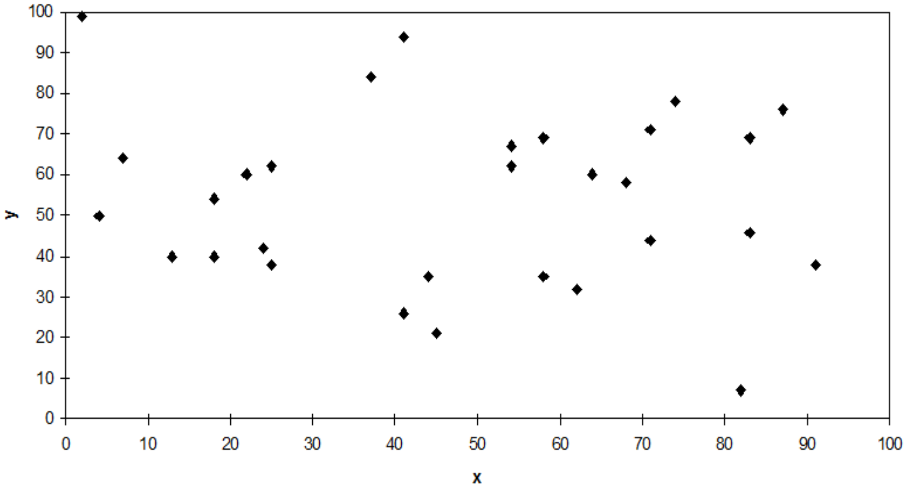
Child2 -> NewList3 ……

Child3 -> NewList4 …….

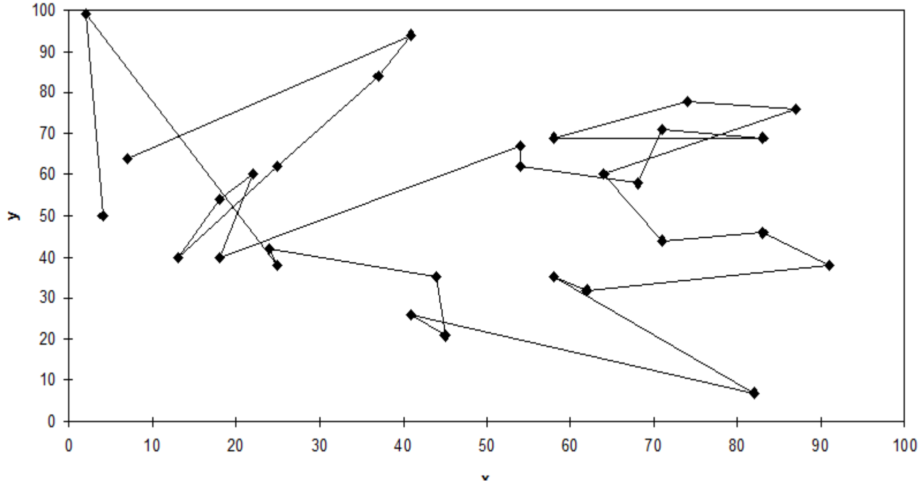
Repeat **Selection**, **Crossover**, and **Mutation** to produce two more child solutions

TSP Illustration– 30 cities

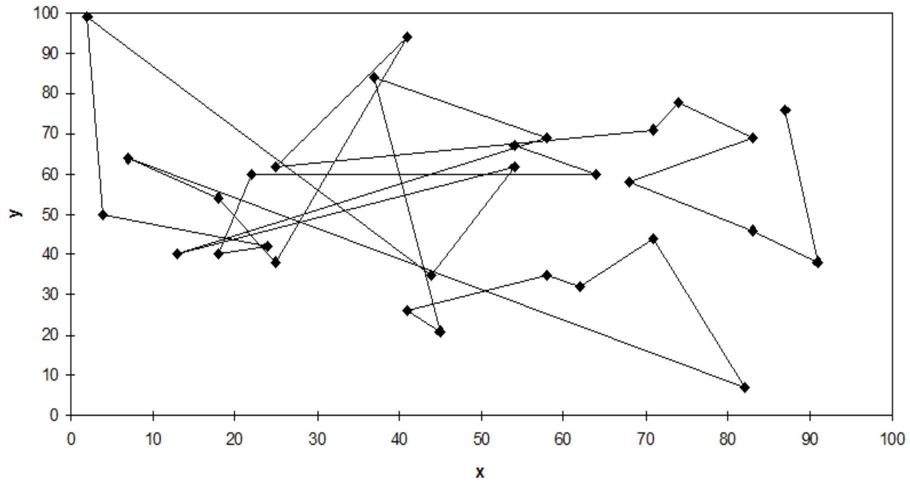
DEMO



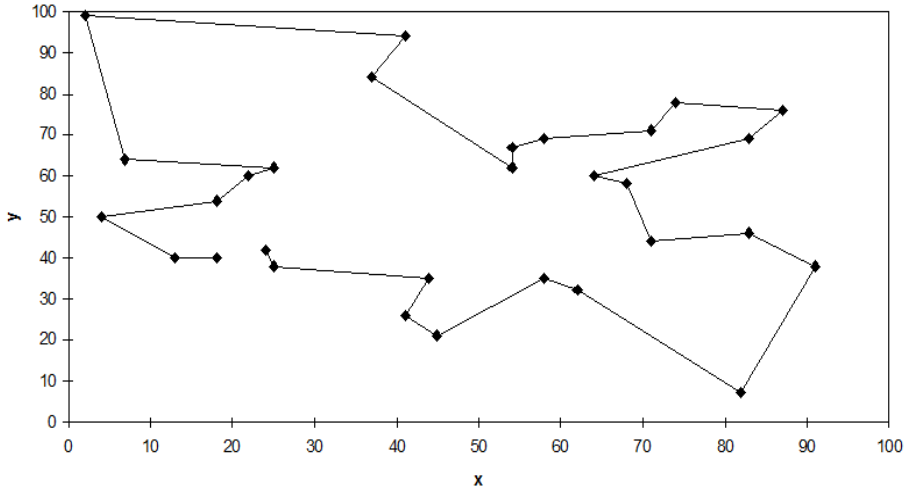
Generation 0



Generation 10



Generation 5



Generation 20

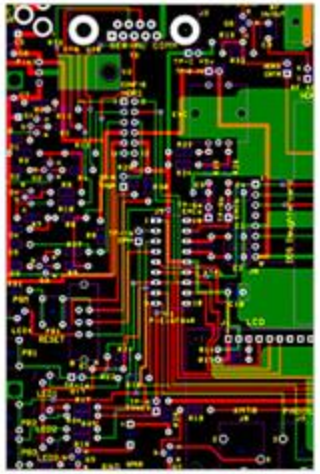
Diagram shows the best solution in the population at corresponding generation

Wendy Williams. Genetic Algorithms: A Tutorial

Practical Applications of TSP

- Finding a tour of a large number of cities in TSP is the same as many important "real world" problems.
  - **Transportation**: school bus routes, service calls, delivering meals, ...
  - **Manufacturing**: an industrial robot that drills holes in printed circuit boards
  - **VLSI (microchip) layout**
  - **Communication**: planning new telecommunication networks

For many of these problems n (the number of "cities") can be 1,000 or more



Outline

- Travel Salesman Problem
- **Machine Learning Optimization**

Feature Selection using Genetic Algorithm

- We will the **Boston Housing** dataset in a **regression task** of predicting house prices.
  - 13 numeric and categorical variables (predictors)
  - Label: price of a house.

Code	Description
CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	nitric oxides concentration (parts per 10 million)
RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940
DIS	weighted distances to five Boston employment centres
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per $10,000
PTRATIO	pupil-teacher ratio by town
B	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT	% lower status of the population
MEDV	Median value of owner-occupied homes in $1000's

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

https://medium.com/@yharsh800/boston-housing-linear-regression-robust-regression-9be52132def4

Regression Problem

- Load the Data set and build a regression model using all 13 features

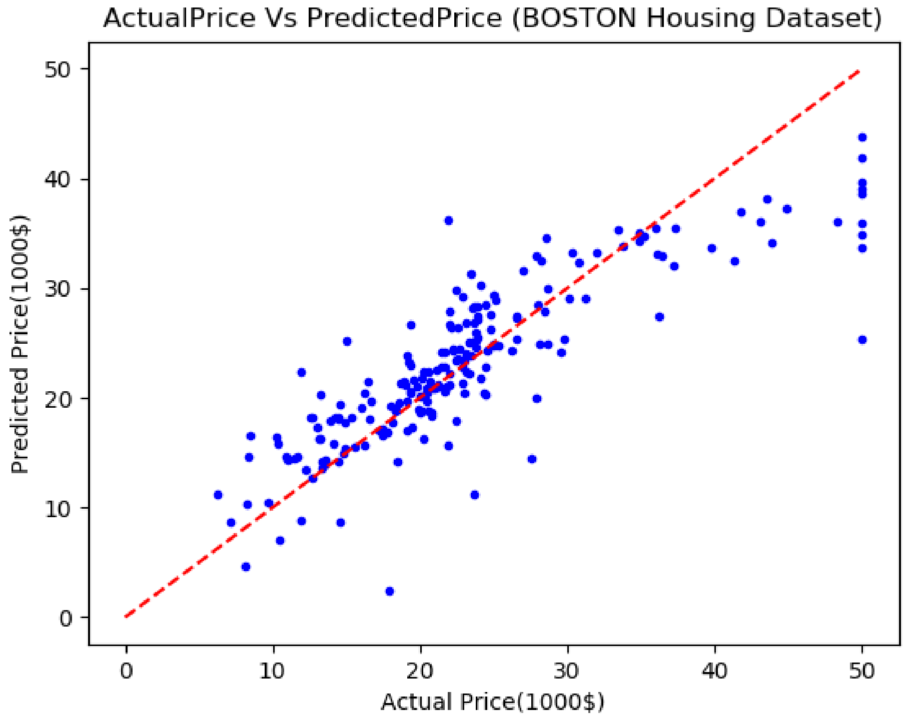
```
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression

dataset = load_boston()
X, y = dataset.data, dataset.target

est = LinearRegression()
score = -1.0 * cross_val_score(est,
                               X,
                               y,
                               cv=5,
                               scoring="neg_mean_squared_error")

print("CV MSE before feature selection: {:.2f}".format(np.mean(score)))
```

```
CV MSE before feature selection: 37.13
```



This error can be reduced with proper feature selection algorithm

Feature Selection using GA

- We can do feature selection be trying all possible combination of the 13 features (*the expensive way*).
- Alternatively, we can use **genetic algorithm** to perform feature selection (*the quick way*) with good results.

Each feature is called a **gene**

feature included	feature excluded
1	0

Binary Representation

A list of 13 genes (features) form a **chromosome**

1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	0	1	1	1	0	1	1	1

A collection of different feature subsets form a **Population**

1	1	0	1	1	1	0	1	1	1	0	1	1	1
2	0	1	1	0	1	1	0	1	1	0	0	1	1
						⋮							
n-1	1	1	0	1	1	0	1	1	1	1	1	1	0
n	1	0	0	1	1	1	1	0	0	0	1	1	1

http://dkopczyk.quantee.co.uk/genetic-algorithm/

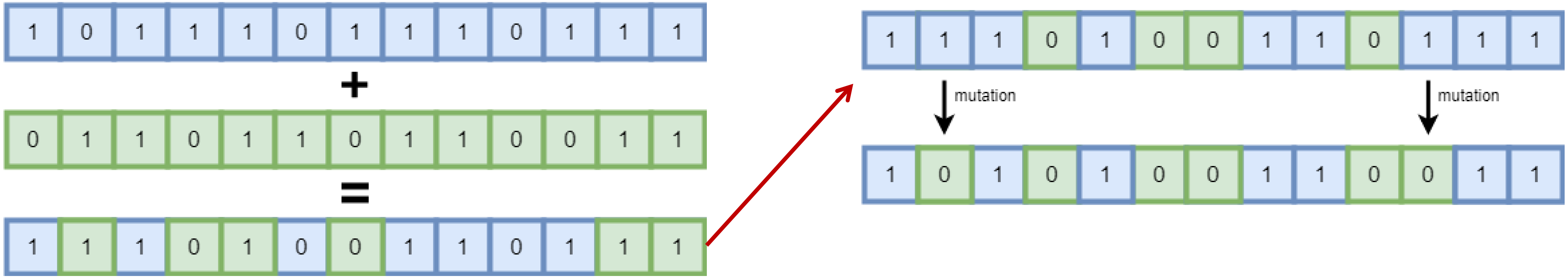
Feature Selection using GA (con.t)

Fitness Function: minimizes the Cross Validation – Means Squared Error (CV-MSE)

Selection: select *n* number of best chromosomes (subsets of features) according to CV-MSE scores, so that our population is moving towards the best solution

Crossover: randomly mix of two individuals.

Mutation: randomly flip the value of gen

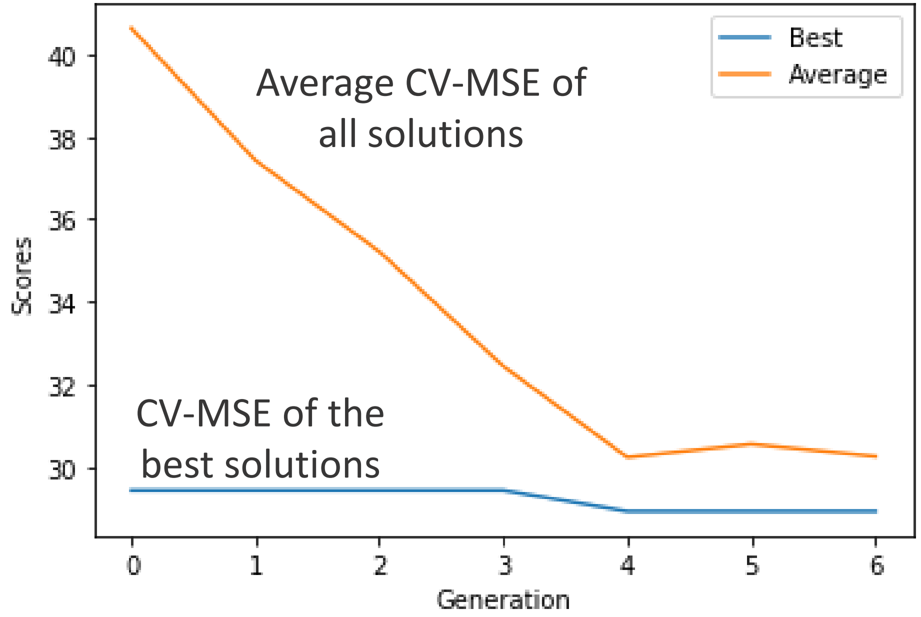


Feature Selection using GA (con.t)

- The genetic operations **selection, crossover** and **mutation** are repeated so that each population should become better and better in terms of the CV scores

```
sel = GeneticSelector(estimator=LinearRegression(),
                      n_gen=7,
                      size=200,
                      n_best=40,
                      n_rand=40,
                      n_children=5,
                      mutation_rate=0.05)
sel.fit(X, y)
sel.plot_scores()
score = -1.0 * cross_val_score(est, X[:,sel.support_],
                               y,
                               cv=5,
                               scoring="neg_mean_squared_error")

print("CV MSE after feature selection: {:.2f}".format(np.mean(score)))
```




```
CV MSE after feature selection: 28.92
```

```
CV MSE before : 37.13
```

We can see after **4** generation the optimizer converged.

Feature Selection using GA (con.t)

- We can extract the best feature sets after running the selection algorithm

```
print('Select features are: ', features[sel.chromosomes_best[4]])
print('Score: ', sel.scores_best[4])
```

```
Select features are:  ['ZN' 'CHAS' 'NOX' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'LSTAT']
Score:  28.922519544512955
```

- Consideration when implement GA in practices:
  - ***representation, population size***
  - ***fitness function***
  - ***crossover operators, mutation rate***
  - ***selection, deletion policies***
  - ***termination Criteria***
  - ***performance, scalability***

Question: *'If GAs are so smart, why ain't they rich?'*

Answer: *'Genetic algorithms **are** rich - rich in application across a large and growing number of disciplines.'*

- David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*

In this lecture, we have covered:

- Introduction to the concepts of evolutionary computation and GA
- Basic terminology and operations of GA
- Business Application GA with TSP case study.
- Machine Learning Application of GA with Feature Selection case study.

Summary