IE440 FINAL22–Murat Ozturk 2019402093

# Traveling Salesman Problem

The traveling salesman problem (TSP) is a classic problem in computer science and operations research. It involves finding the shortest possible route that visits a given set of locations and returns to the starting point.

# Euclidean Traveling Salesman Problem

The Euclidean traveling salesman problem (ETSP) is a variant of the traveling salesman problem (TSP) where the distance between two locations is the Euclidean distance, which is the straight-line distance between the two points. The straight-line distance between two points is calculated using the Euclidean distance formula, which is based on the Pythagorean theorem.

# ETSP using Self Organizing Map

Self-organizing maps (SOMs) are a type of artificial neural network that can be used for clustering data etc.

Self-organizing maps can also be used to solve the ETSP problem. If the algorithm works as intended, the weights of the nodes will represent the route passing through the points. It is important to note that the solution obtained using this approach may not necessarily be the shortest possible route.

# The Kernels

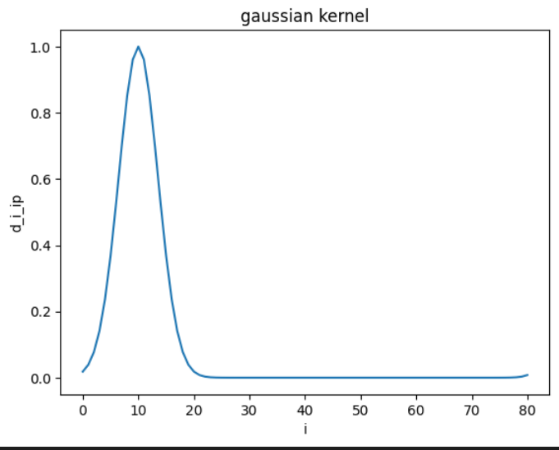
First the kernels are defined.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, cihaz, metre içeren bir resim

Açıklama otomatik olarak oluşturuldu

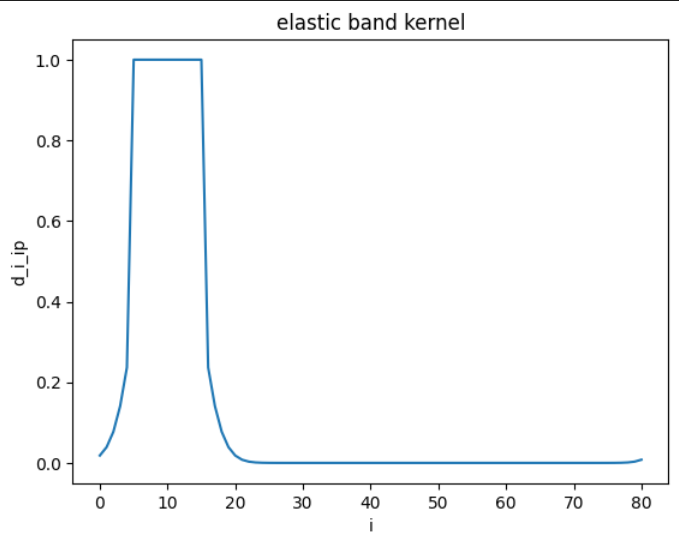
The input to these kernels will be diip. Below, is the example as the gaussian kernel.

metin, ölçü aleti, cihaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

In the algorithm to be provided, the diip values will be given as a vector d\_i which contains values d\_0\_ip, d\_1\_ip, d\_2\_ip, … d\_M\_ip. The output of the kernels will be like the plot, where the peak is the best matching unit’s index. In the figure, there are 81 neurons.

For the elastic band kernel, the plot will be like figure below.



# The Algorithm

## The documentation of the parameters:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## How the algorithm works:

First some values are initialized.

The default for M is number of cities.

Default for sigma is M/2.

Weights are initialized randomly between 0 and 1

W\_prev holds, previous weights. This will be explained later.

Then the iterations for loop:



Random permutation is created at length of the cities.

r\_order (random \_ order)

Iterate all cities, for loop:

metin, ekran, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

For each city, using the order in r\_order, the distances to each neuron weights are calculated. Then the smallest distance, thus the best matching unit (BMU) is found. ‘i\_p’ is the index of the BMU.

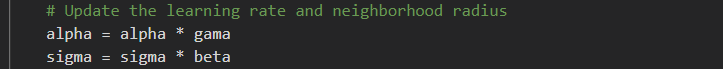
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Order\_distance is diip. d\_i vector is calculated. The kernel vector is calculated using the chosen kernel function. Then 1 dimensional kernel vector is reshaped into (M, 1) 2 dimension to be broadcasted.

Weights are updated accordingly.

End for loop, p in r\_order



The learning rate and neighborhood radius are updated.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

If track is set to True, then the process of training can be tracked with plots.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Break for loop if the changes in all weights are too small, or the learning rate is too small.

If not, record the current weights as W\_prev to compare in the next iteration.

End for loop, it in range(max\_iter)

Finally return the results.

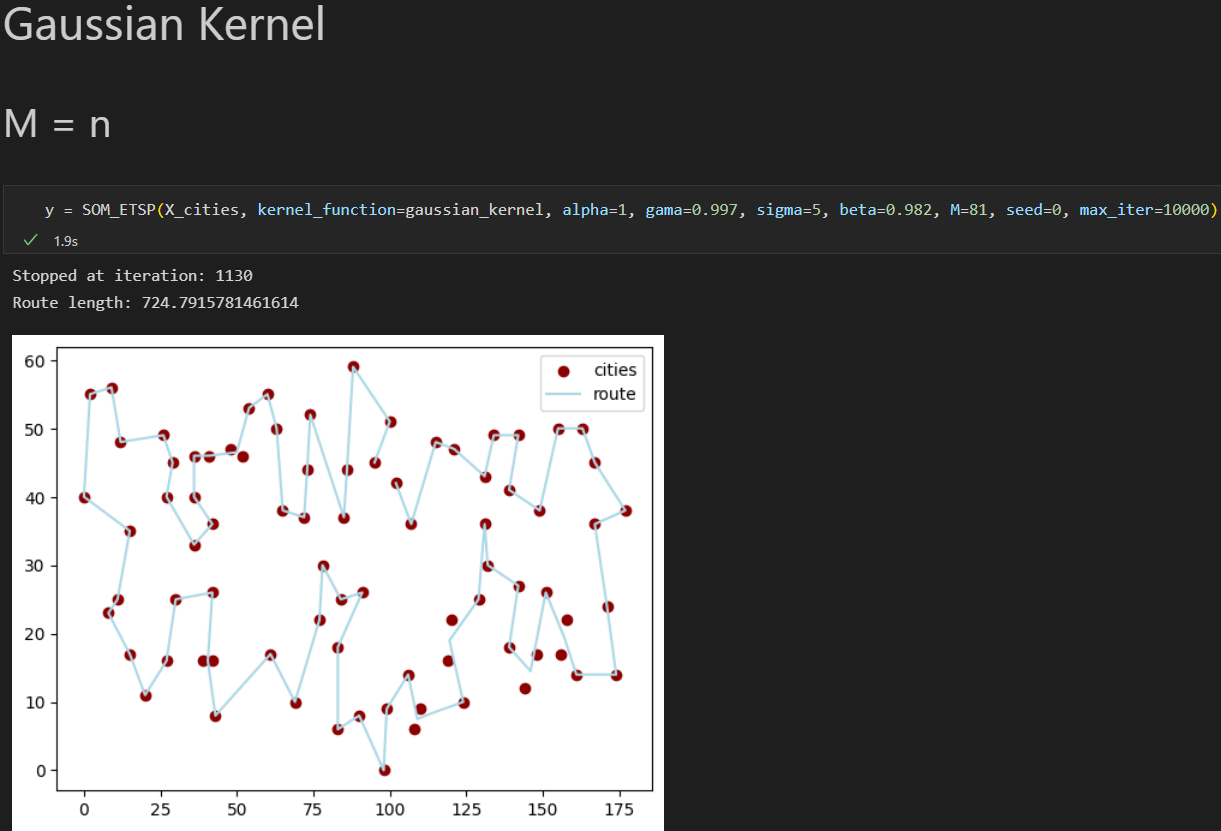
## All the algorithm in one picture:

metin içeren bir resim

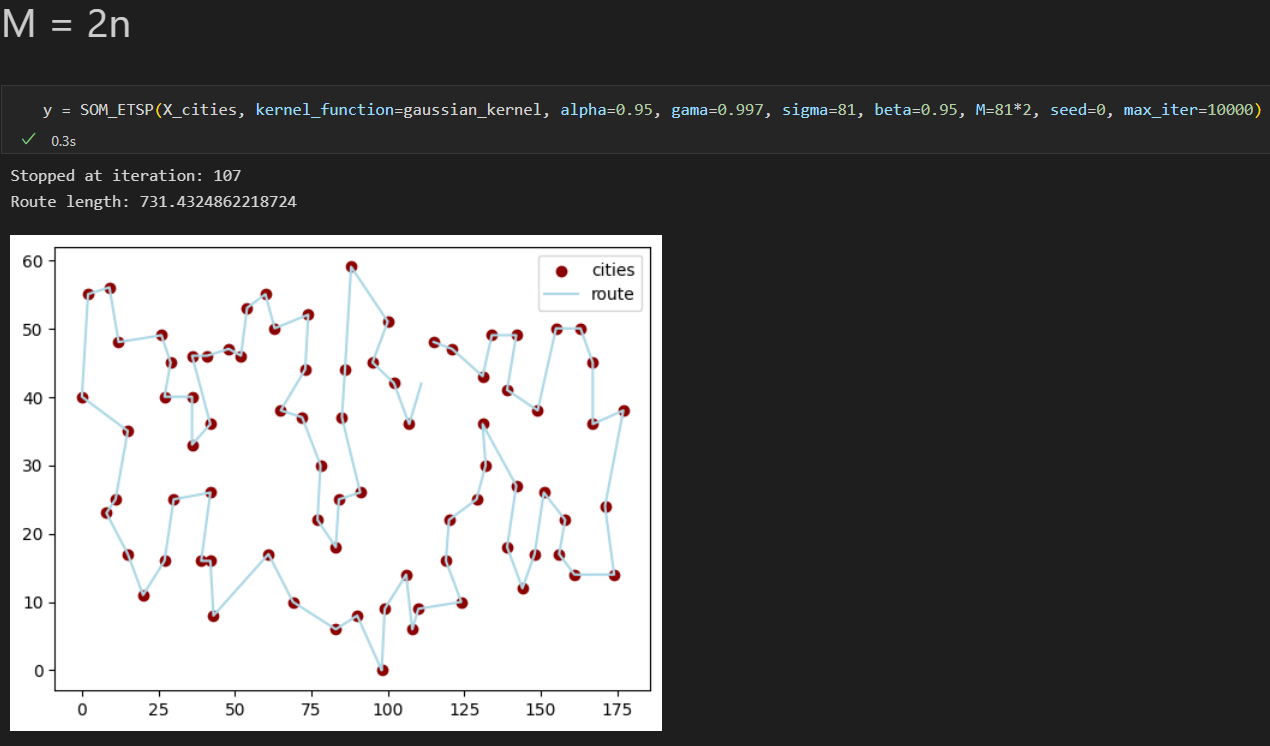
Açıklama otomatik olarak oluşturuldu

# Outputs

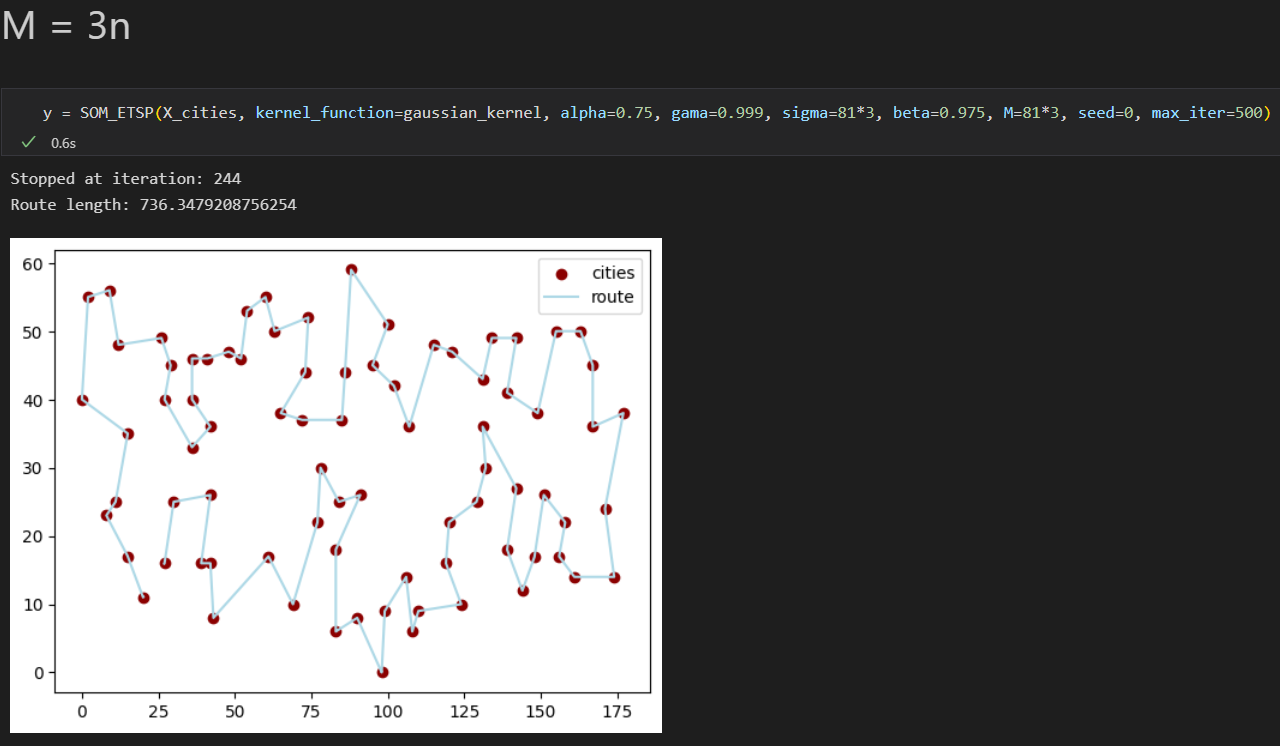
## Gaussian Kernel



The result isn’t satisfying. There are some cities that are not covered by the route.



All cities are covered, this can be used as a solution.

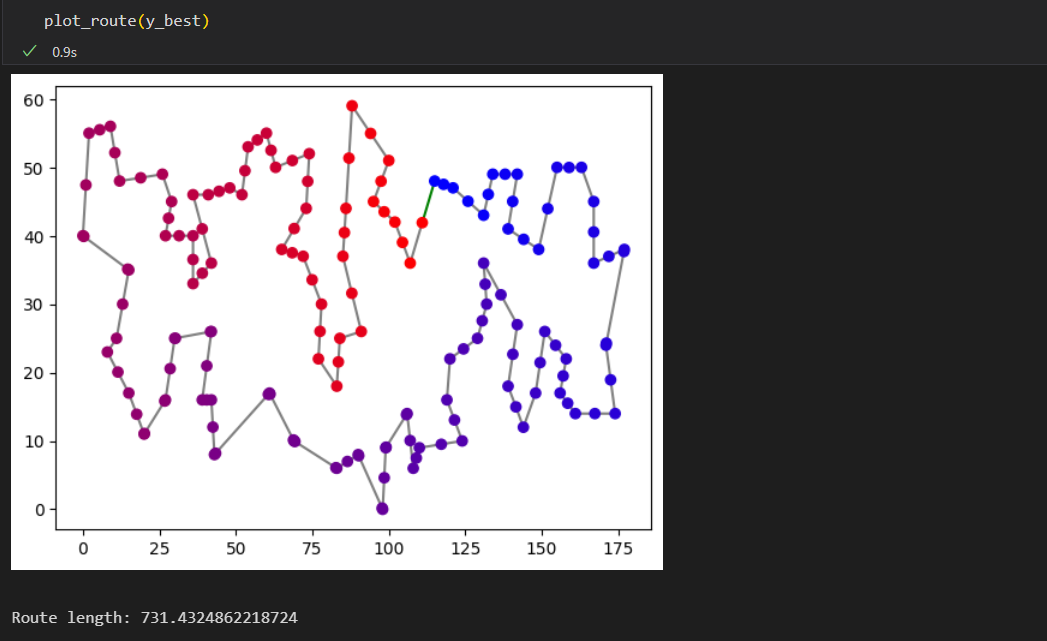


All cities are covered, this can be used as a solution.

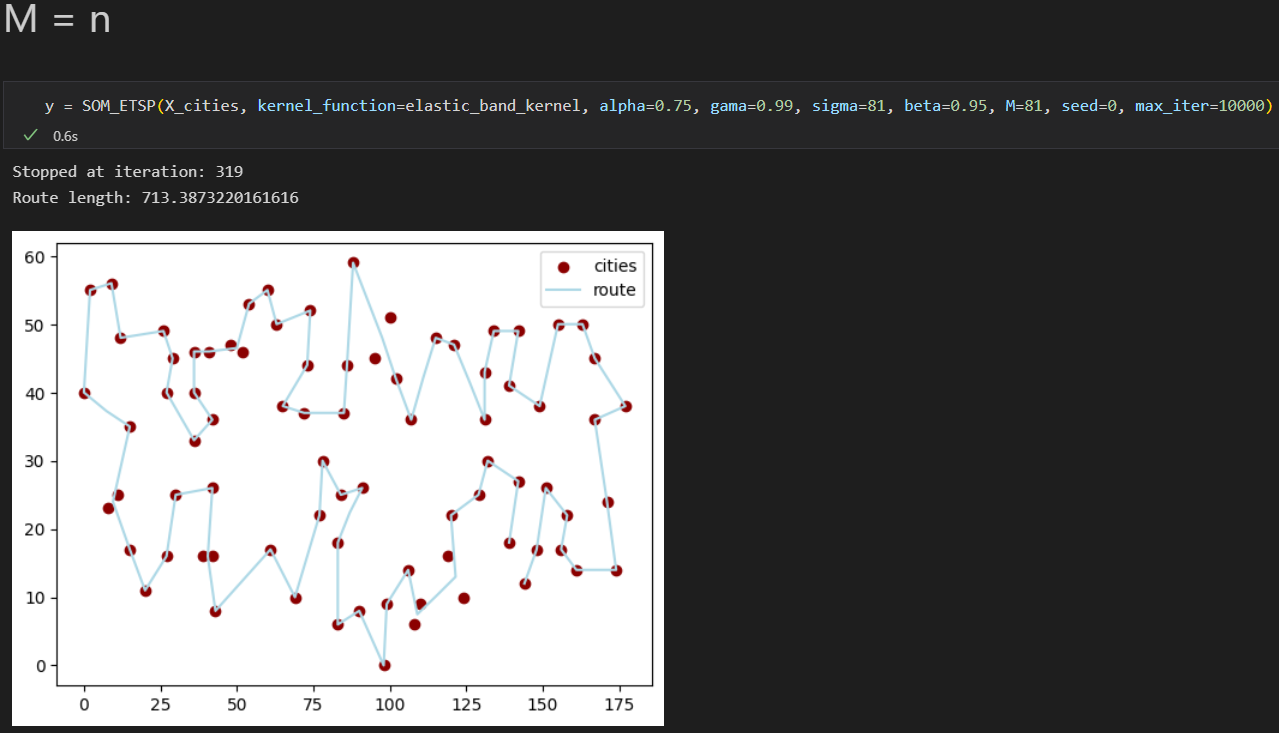
### Best Solution



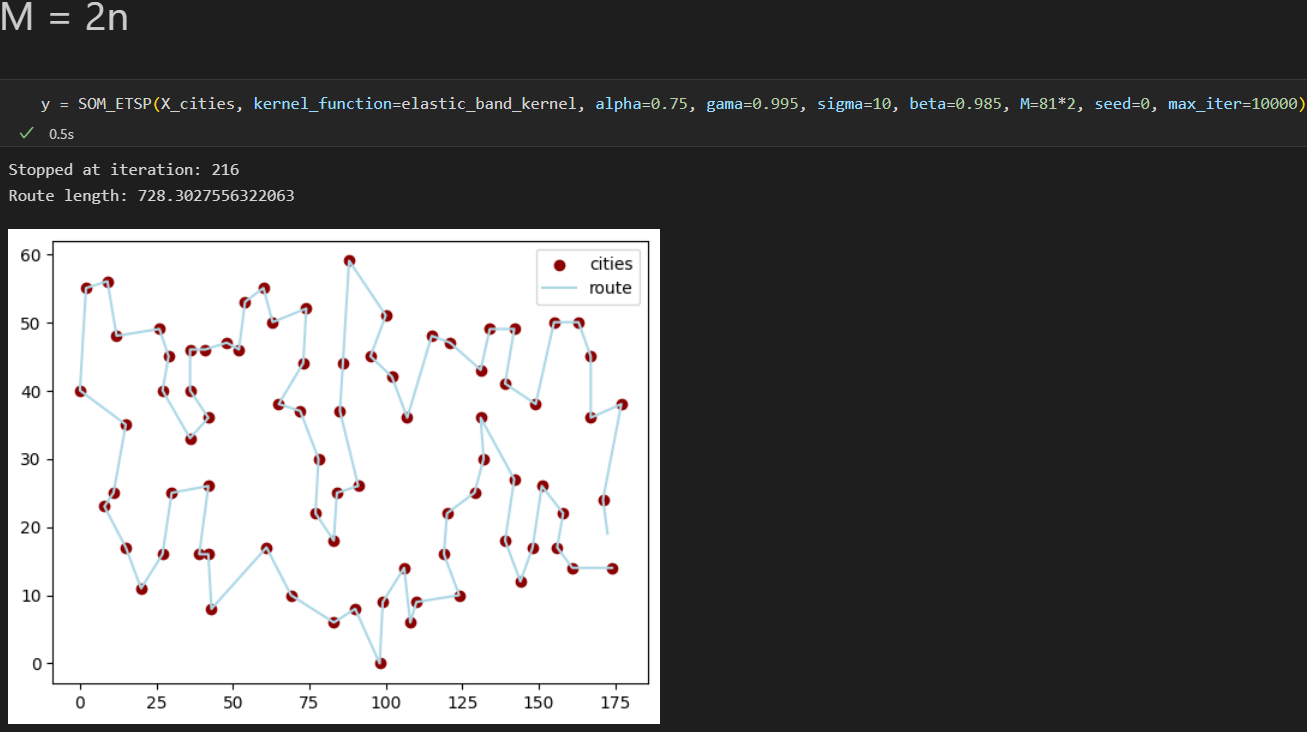
Route starts from green line, blue to red.



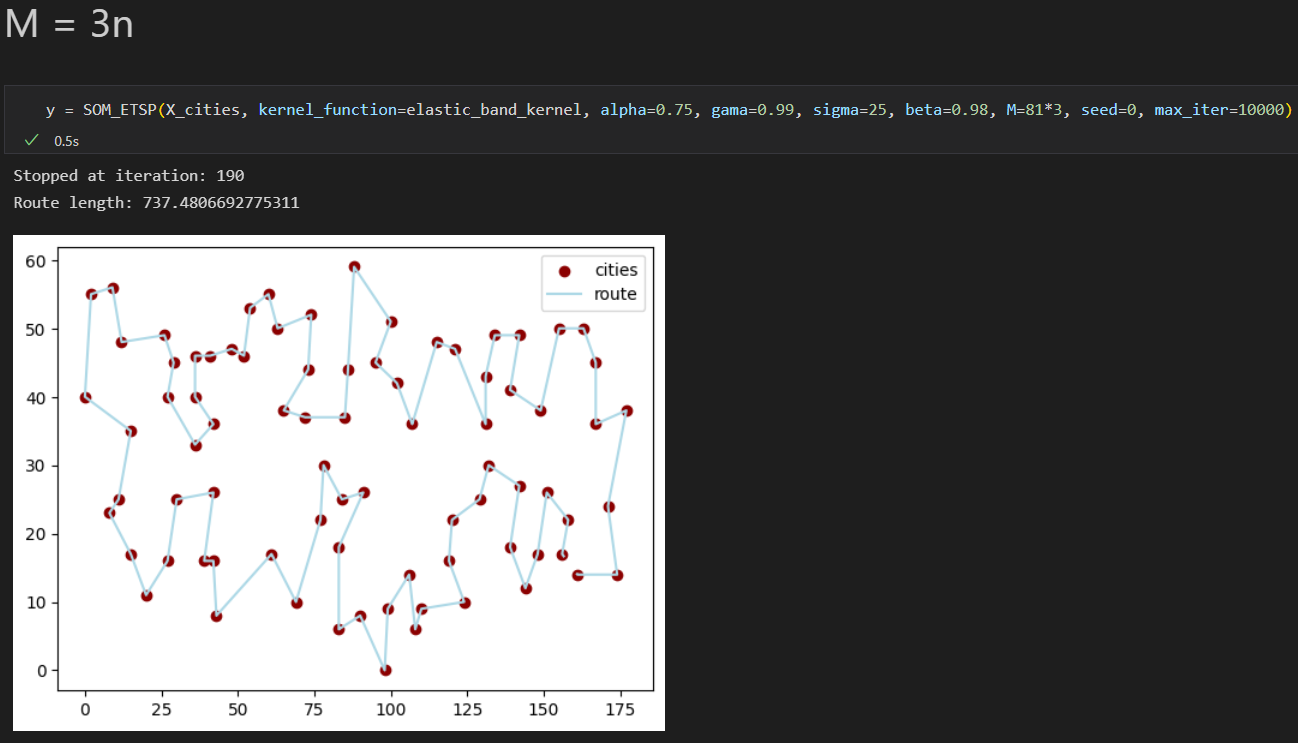
## Elastic Band Kernel



The result isn’t satisfying. There are some cities that are not covered by the route.



All cities are covered, this can be used as a solution.



All cities are covered, this can be used as a solution.

### Best Solution



Route starts from green line, blue to red.

