MATHEMATICS FOR INTELLIGENT SYSTEMS (MIS)

Project on Travelling Salesman Problem (TSP)

Shivam Basia
UF ID:7838-4123

Problem Statement

We have been given the x-y coordinates of each city and we want to calculate the minimum length hamiltian cycle for all this cities.Since,this is an NP-Hard problem we try to find the most optimal value by minimizing the objective function.

$$E_{\mathrm{TSP}}(P,Y)=\sum_{i=1}^{N}\sum_{a=1}^{M}P_{ia}\|x_i-y_a\|_2^2+\frac{\kappa}{2}\sum_{a=1}^{M}\|y_a-y_{a\oplus 1}\|_2^2.$$

This is an example of elastic net objective function using softmax non-linearity.

So,to start we start with a uniform distribution in the range of $[0,1]^2$.

We initially normalize the cities x-y coordinate to gather a uniform distribution in the range of [0,1].

We have taken the number of hidden cities factor as 3, so in an 100 city TSP total no of hidden cities will be 300 which will be the number of hidden cities.

For calculating Y, We start with the centroid and initialize the hidden cities in the range(0,2*pi*radius) where radius is taken as 0.1 along with some noise of 0.001.

We gradually try to minimize our objective function in each of the 2000 iterations we have taken.
We are also gradually updating k as k/gamma with the starting k value as 0.2 and the minimal value been 0.01.
Also,we are increasing beta as b*gamma with the starting value of 10 and the maximal value as 500.
The gamma value is taken as 1.05.

The value of beta and kappa are updated at an interval of every 25 iterations.

We use the below softmax nonlinearity function for our TSP problem:

$$P_{ia}=\frac{\exp\{-\beta\|x_i-y_a\|_2^2\}}{\sum_{b=1}^{M}\exp\{-\beta\|x_i-y_b\|_2^2\}}$$

Which we alternate with the equation of Y as below:

$$Y=(\kappa L+D)^{-1}P^{T}X$$

And we keep alternating and updating P until convergence.

For the first problem,we generate a uniform random distribution 100 cities coordinates and execute our Elastic net algorithm.

The source code is given on the next two pages(if we want to generate random distribution and not the TSP48 then uncomment line 16 and comment 9-14) :

Source code

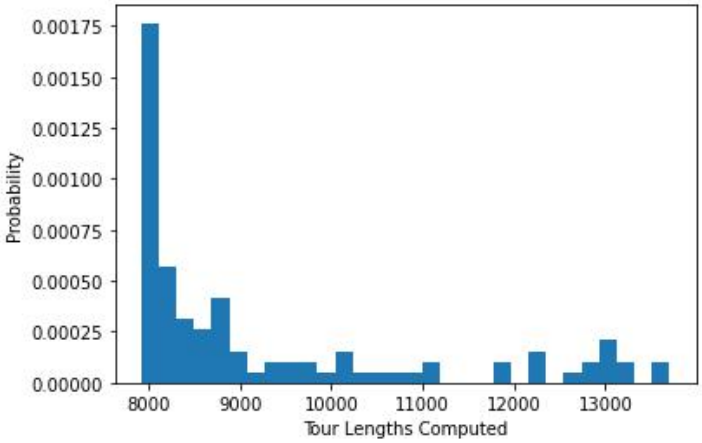
```
import math
import numpy
import itertools
from matplotlib.collections import LineCollection
import matplotlib.pyplot as plt
import statistics
import scipy.linalg as spla

file = open('TSP48.txt')
original_cities = []
for line in file:
	x, y = line.strip().split()
	original_cities.append((int(x), int(y)))
original_cities = numpy.array(original_cities, dtype=float)
N=48
#original_cities =numpy.random.uniform(0, 1000, size=(100, 2))
cities =(original_cities - (original_cities.min(axis=0))) / ((original_cities.max(axis=0)) -
(original_cities.min(axis=0)))

beta=20
gamma=1.05
k_update_period=25
max_num_iter=2000
MN_factor=3
M=N*MN_factor
radius=0.1
num_iter=0
k=0.2
noise=0.001
theta = numpy.linspace(0, 2 * math.pi,M, False)
centroid = cities.mean(axis=0)
Y = numpy.vstack((numpy.cos(theta), numpy.sin(theta)))
Y *= radius
Y += centroid[:,numpy.newaxis]
Y+=noise
Y = Y.transpose()
def distance(point1, point2):
	x1, y1 = point1
	x2, y2 = point2
	return int(math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2) + 0.5)
trial_edges=[]
trial_permutation=[]
trial_length=[]
loss=[]
while    num_iter    < max_num_iter:
	num_iter += 1
	if (num_iter % k_update_period) == 0:
		k = max(0.01, k/gamma)
		beta=min(500,beta*gamma)
```

```
        delta = cities[:,numpy.newaxis] - Y
        dist2 = (delta ** 2).sum(axis=2)
        worst_dist = numpy.sqrt(dist2.min(axis=1).max())
        Loss.append(worst_dist)
        P = numpy.exp(-dist2 * beta )
        P /= P.sum(axis=1)[:,numpy.newaxis]
        D=numpy.array([numpy.dot(P[:,i],delta[:,i]) for i in range(M)])
        L=numpy.concatenate(([Y[1] - 2 * Y[0] + Y[M - 1]],
        [(Y[i+1]- 2 * Y[i]+ Y[i-1]) for i in range(1, M - 1)],
        [Y[0]- 2 * Y[M - 1]+ Y[M - 2]]))
        Y +=    D +    k * L
        if num_iter%200==0:
                c,d=itertools.tee(itertools.chain(range(len(Y)),[0]))
                next(d, None)
                elastic_edges=list(zip(c,d))
                figure = plt.figure()
                figure.gca().scatter(cities[:,0], cities[:,1], s=15, color='black')
                elastic_collection = LineCollection(Y[elastic_edges], edgecolor='green')
                figure.gca().add_collection(elastic_collection)
                figure.gca().scatter(Y[:,0], Y[:,1], s=3, color='green')
                plt.show()
        if num_iter%50==0:
                neuron_city_pair = []
                for i in range(dist2.shape[0]):
                        city = dist2.min(axis=1).argmin()
                        neuron = dist2[city].argmin()
                        dist2[city] = numpy.inf
                        dist2[:,neuron] = numpy.inf
                        neuron_city_pair.append((neuron, city))
                        neuron_city_pair.sort(key=lambda x: x[0])
                        permutation=[x[1] for x in neuron_city_pair]
                a,b=itertools.tee(itertools.chain(permutation, [permutation[0]]))
                next(b, None)
                edges=list(zip(a,b))
                length = sum([distance(original_cities[src_dst[0]],original_cities[src_dst[1]]) for src_dst in edges])
                trial_edges.append(edges)
                trial_permutation.append(permutation)
                trial_length.append(length)
shortest_length=min(trial_length)
shortest_iteration = trial_length.index(shortest_length)
shortest_permutation=trial_permutation[shortest_iteration]
shortest_edges=trial_edges[shortest_iteration]
print("Shortest Length:"+str(shortest_length))
print("Shortest Permutation:"+str(shortest_permutation))
figure = plt.figure()
figure.gca().scatter(cities[:,0], cities[:,1], s=15, color='black')
edge_collection = LineCollection(cities[numpy.array(shortest_edges)], edgecolor='blue')
figure.gca().add_collection(edge_collection)
plt.show()
plt.plot(loss)
plt.show()
```

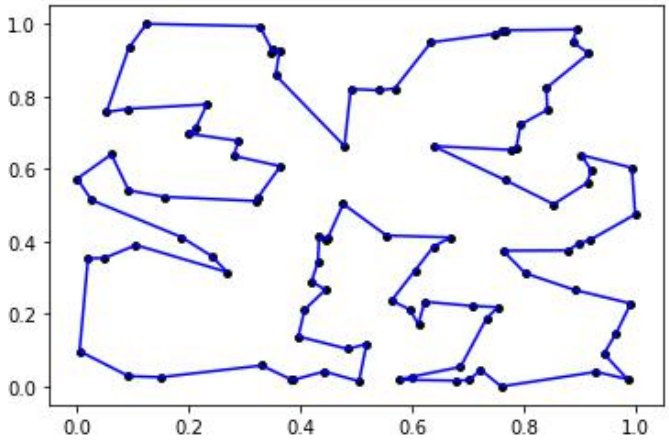
Observations on TSP100

The average length,shortest length,median length,longest length of the TSP computed for TSP100 for 100 trials is given below along with the visualization:

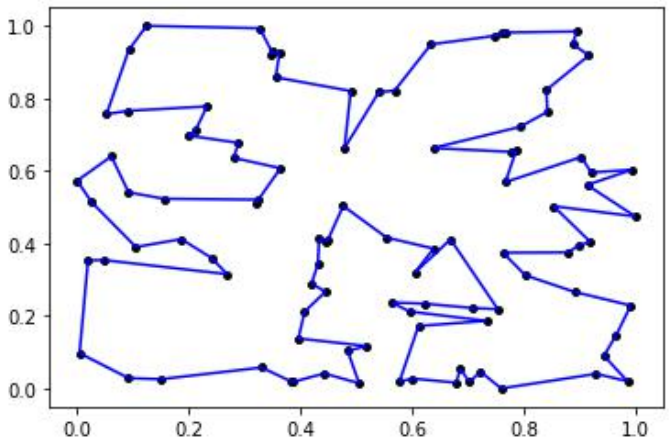
```
Average Length:9.26996
Shortest Length:7.92
Median Length:8.4055
Longest Length:13.699
```



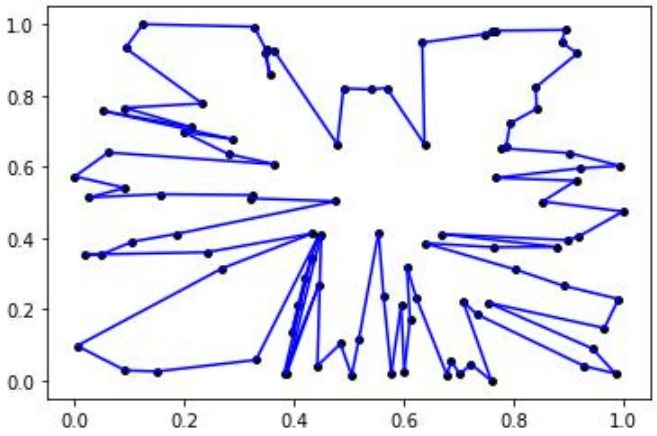
Histogram plot of 100 trials for TSP100(The tour lengths are multiplied with 1000 for better plot visualization)



Shortest Length Tour of TSP100 City(Total Length:7.92)



Median Length Tour of TSP100 City(Total Length:8.4055)

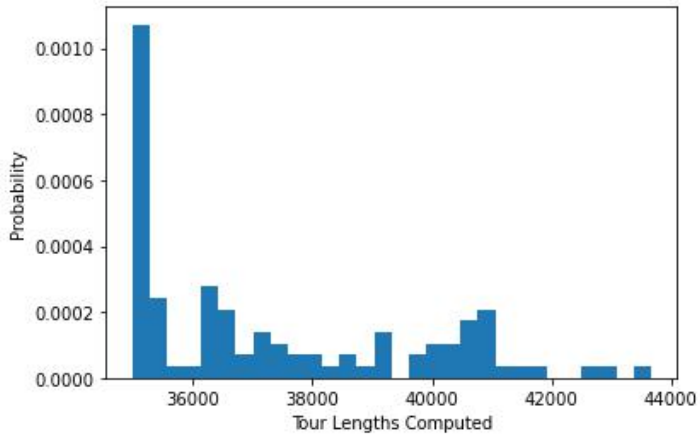


Longest Length Tour of TSP100 City(Total Length:13.699)

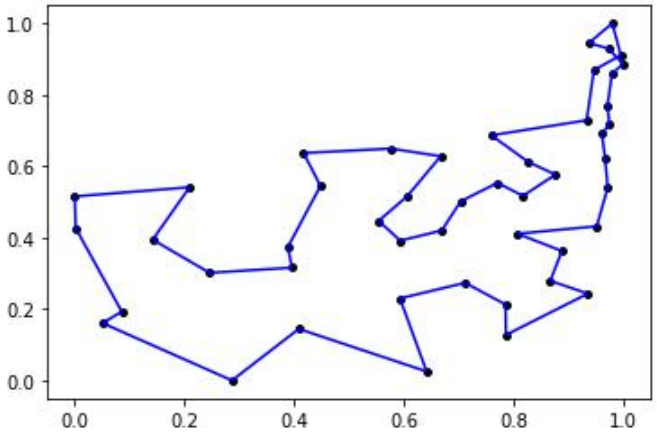
Observations of Elastic Net Implementation on 48 city TSP

The average length,shortest length,median length,longest length were computed for 100 trials with gradual hyper-parameter tuning for the 48 city TSP Problem as well.The visualization is given below as well. The actual minimum tour length is actually 33523 according to the website and our shortest length produced a result of 34984 which is only 4.4% greater than the optimal one.So,we can conclude that our algorithm provided a good solution .

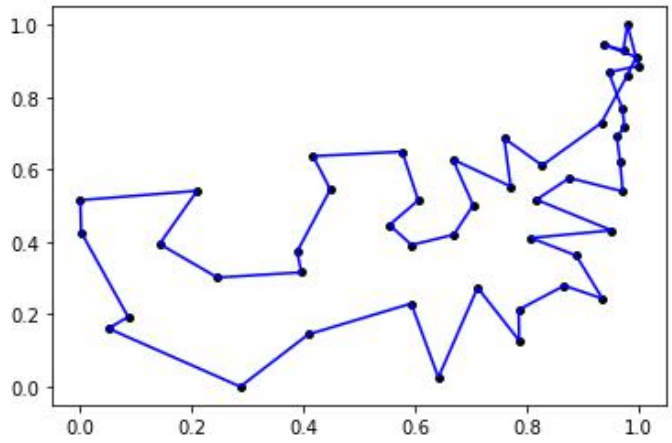
```
Average Length:37398.35
Shortest Length:34984
Median Length:36518.0
Longest Length:43660
```



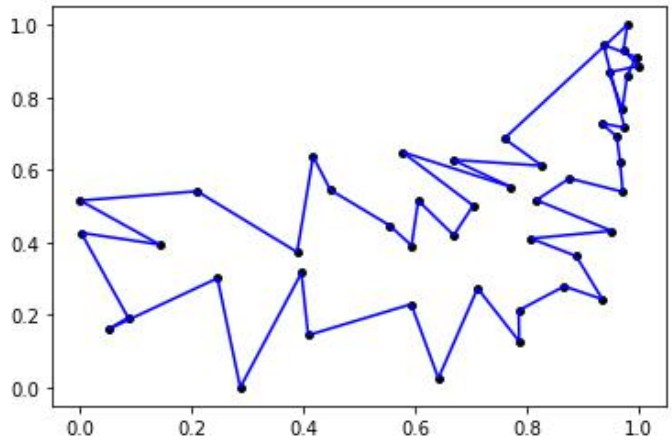
Histogram plot of 100 trials for TSP48



Shortest Length Tour of TSP48 City(Total Length:34984)



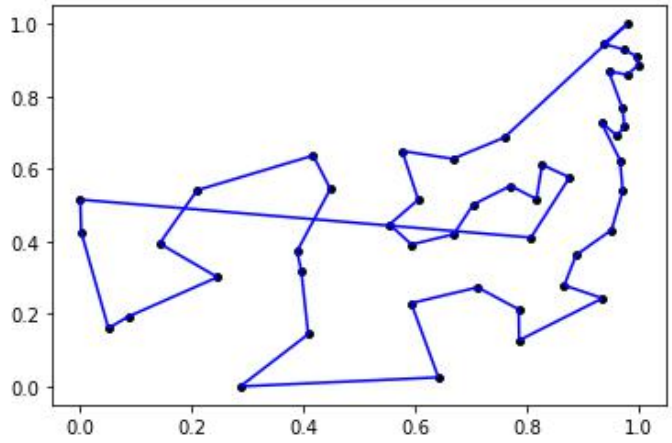
Median Length Tour of TSP48 City(Total Length:36518)



Longest Length Tour of TSP48 City(Total Length:43660)

Comparison of MST Heuristic and Elastic Net

For MST Heuristic Comparison,we compared elastic net TSP with the MST Heuristic TSP of 48 cities.
The MST Heuristic shortest tour length for 48 cities came out to be 40160.368546331825.
The MST Heuristic tour Output plot for 48 cities is given below:



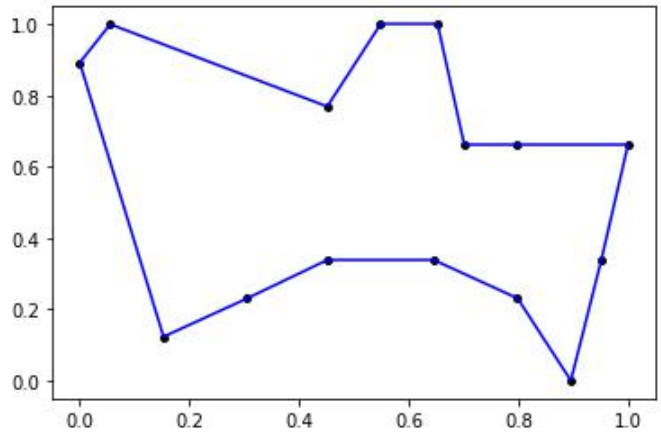
MST Heuristic Solution of TSP 48 City(Tour Length:40160.3685).

We can clearly see that elastic net did better than the MST Heuristic Solution.Also, we know that the MST Heuristic is a 2-approximation algorithm whereas the worst case approximation ratio of our solution for 48 TSP came out to be 43660/33523=1.3 which is better than the MST Heuristic's approximation ratio.

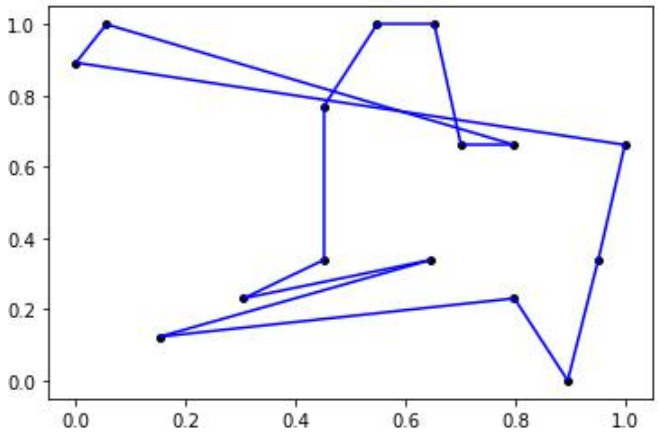
Comparison of Held-Kalp Lower Bound and Elastic Net

For Held-Karp Lower bound, since it is very difficult to output a solution due to the time complexity of it as it is O(2^n*n^2), we tried with the 15 city problem(P01) given in the same website.The Held-Karp solution gave a output of 302 as the total length tour whereas our algorithm gave the length tour as 389.

The visualization is given below:



Tour Length of Elastic Net for 15 city(Tour Length:389)



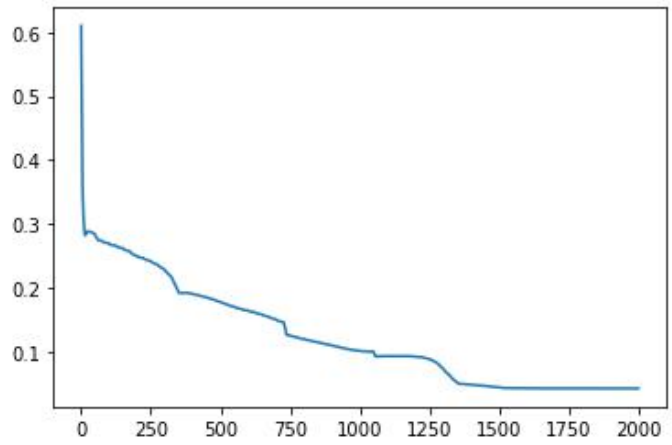
Tour Length of Held-Karp for 15 city(Tour Length:302)

As we can say that Held-Karp solution is obviously much better than elastic net in terms of finding optimal tour length, we will be only able to compute this algorithm with less size as the time complexity is exponential in nature which will allow us most probably 30 cities in a very good computing environment.
So, in case of large number of cities we can say that elastic net provides a good approximation tour length as in this case it was 389/302=1.29 which is quite good in respect of other algorithms.

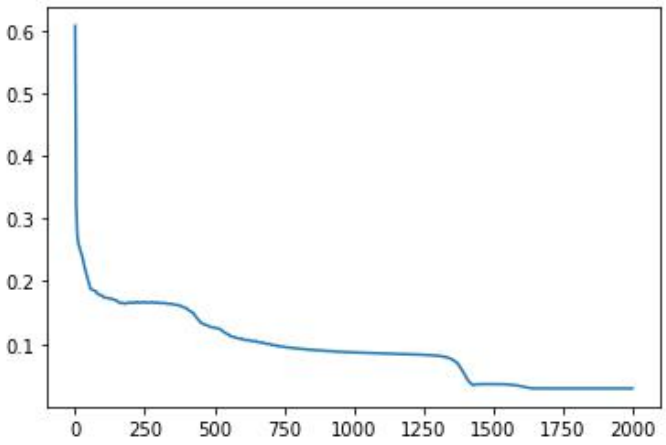
Also we tried checking if the lower bound is $L = K\sqrt{N}$ where 0.765<=K<=0.765+4/N and found that the lower bound seems to satisfy when k is in the range of 0.792<=K<=0.792+4/N.So, the elastic net implementation approximately satisfies the required lower bound.

Convergence plot of the algorithm

Since we implemented the equation (1), the convergence plot obtained in TSP100 and TSP48 is been visualized below.



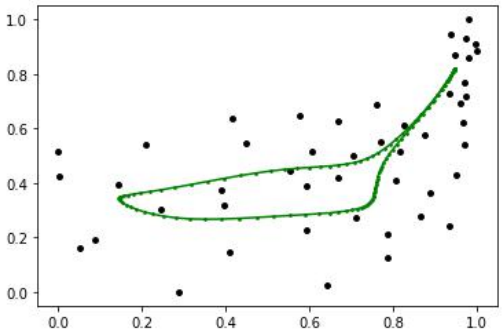
Convergence plot of Equation(1) for TSP100



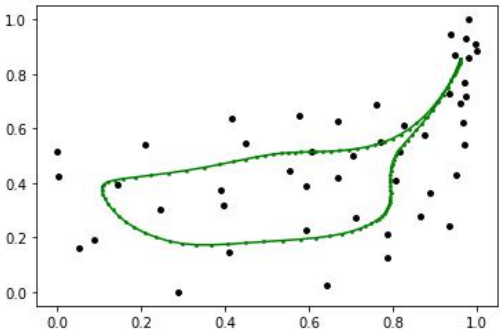
Convergence plot of Equation(1) for TSP48

Also, on the next page are the visualizations for TSP48 and TSP100 which visualizes how the objective function starts like a circle and expands like an elastic net to cut a best distance path to give us an optimal tour length.As we can see the objective continues to converge to a global minima with each iteration.

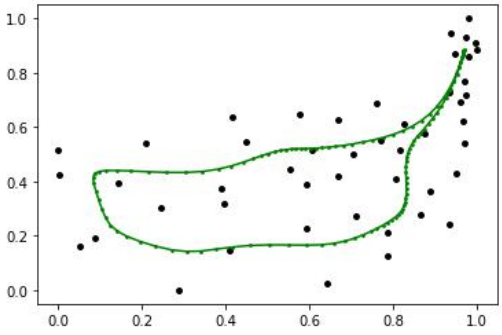
Visualization of Elastic Net of TSP48



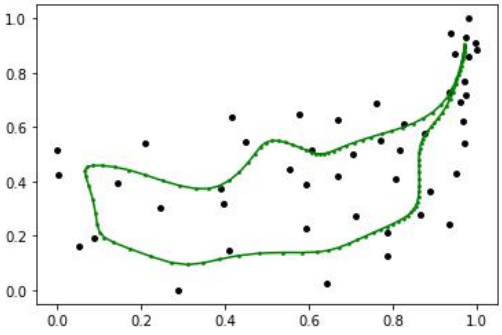
Iteration 200



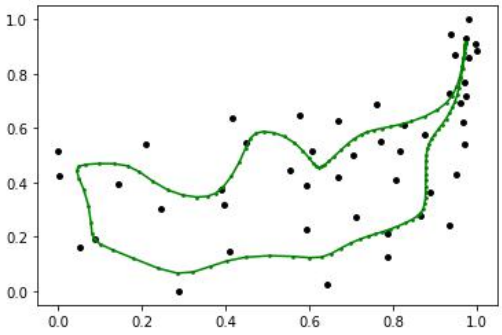
Iteration 400



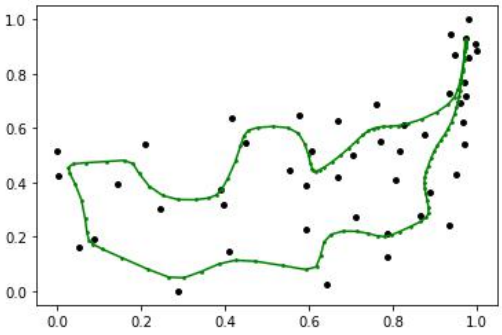
Iteration 600



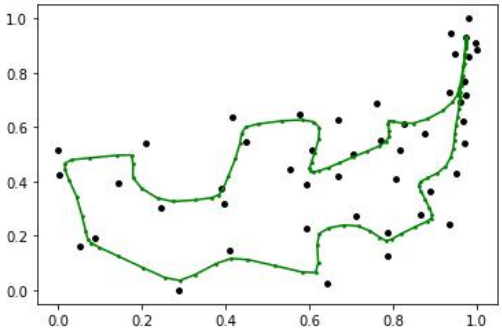
Iteration 800



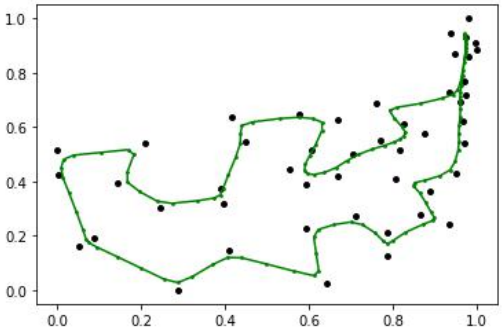
Iteration 1000



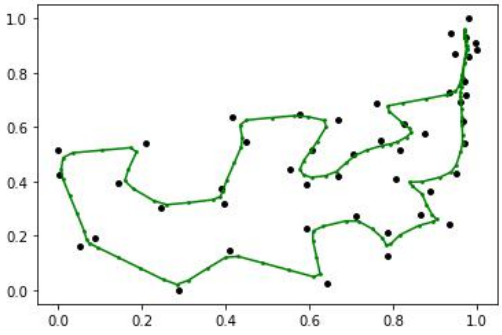
Iteration 1200



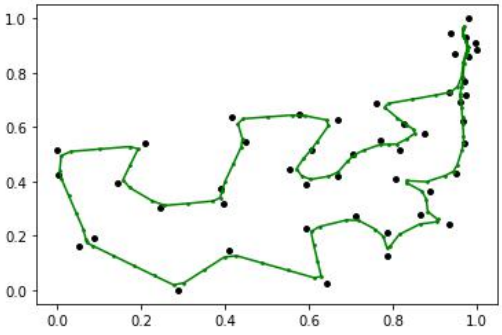
Iteration 1400



Iteration 1600

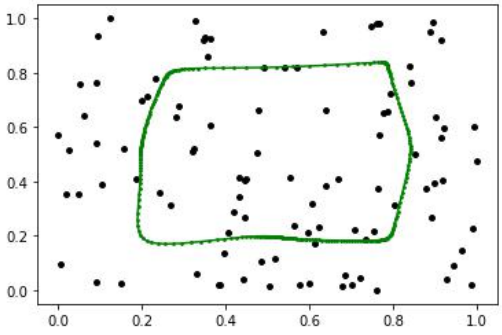


Iteration 1800

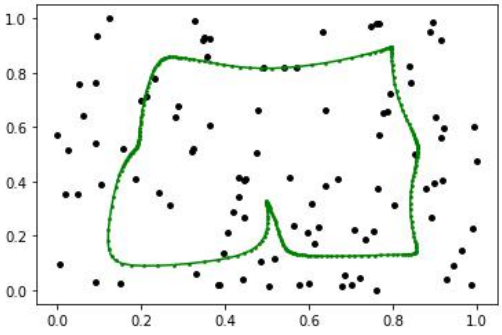


Iteration 2000

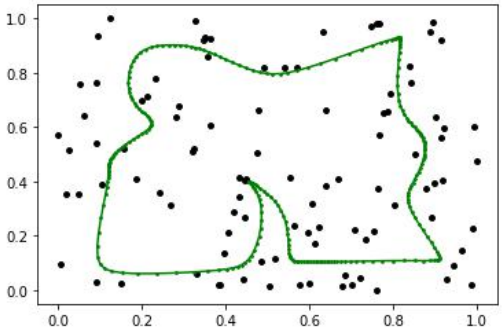
Visualization of Elastic Net of TSP100



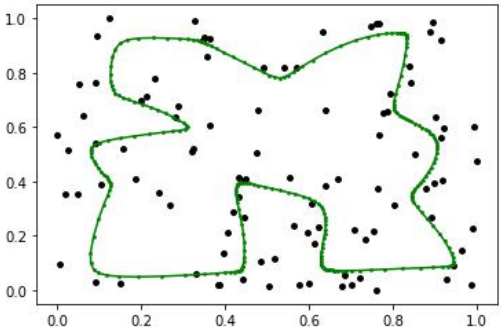
Iteration 200



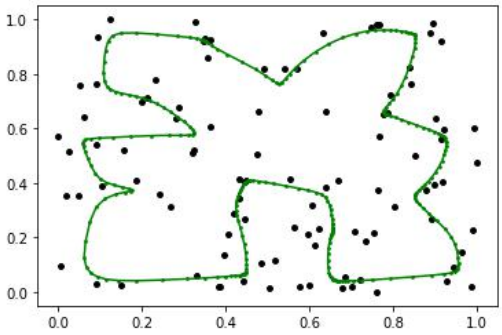
Iteration 400



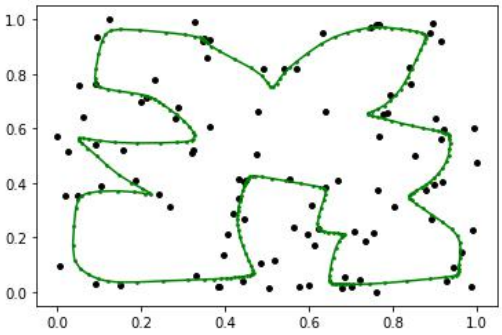
Iteration 600



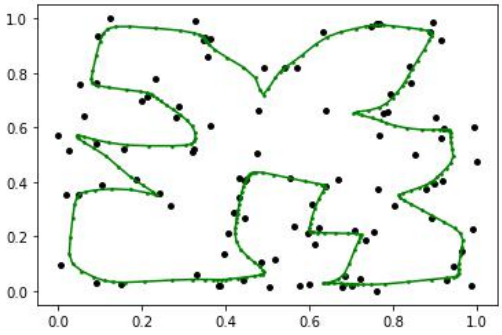
Iteration 800



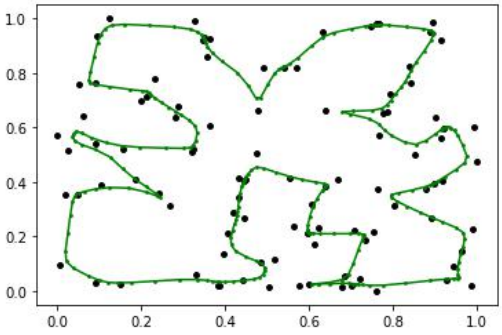
Iteration 1000



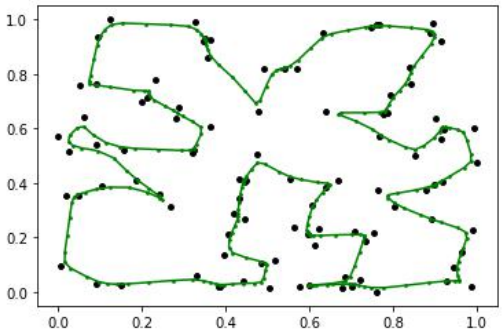
Iteration 1200



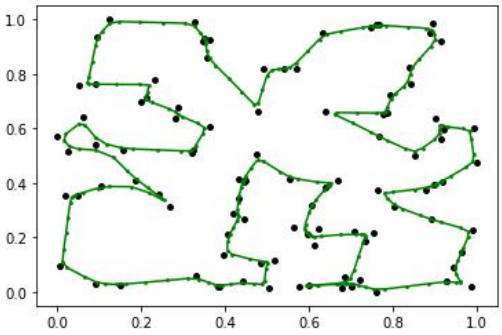
Iteration 1400



Iteration 1600



Iteration 1800



Iteration 2000

Explanation of Free Energy Equivalent Function(1)

We have the initial objective function as

$$E_{\mathrm{TSP}}(P,Y)=\sum_{i=1}^{N}\sum_{a=1}^{M}P_{ia}\|x_i-y_a\|_2^2+\frac{\kappa}{2}\sum_{a=1}^{M}\|y_a-y_{a\oplus 1}\|_2^2.$$

Here,P can be termed as “synaptic potentials”.

k is a free parameter which can be termed as “neuron potentials”.

So,Gibbs distribution here will be

$$\Pr(S=P)=\exp\{-\beta\,\|\,x_i-y_a\,\|_2^2\}$$

The partition function becomes

$$Z=\sum\exp\{-\beta[\sum_{i=1}^{N}\sum_{a=1}^{M}P_{ia}\,\|\,x_i-y_a\,\|_2^2]\}\,.$$

Now, $\Pr(S=P)=\dfrac{\exp\{-\beta\,\|\,x_i-y_a\,\|_2^2\}}{\sum_{b=1}^{M}\exp\{-\beta\,\|\,x_i-y_b\,\|_2^2\}}$

which is the softmax non-linearity.

We know that the expected value of energy function is

$$U=\mathcal{E}\left[-\sum_i T_i s_i\right]=-\sum_i T_i\mathcal{E}[s_i]=-\sum_i T_i\frac{\exp\{\beta T_i\}}{\sum_j\exp\{\beta T_j\}}$$

Here,the expected value of energy is softmax weighted sum of $T_i's$.

So, the Free energy defined as $-\frac{1}{\beta}\log Z$ here is

$$F=-\frac{1}{\beta}\log\sum_i\exp\{BT_i\}$$

Now,from our objective function we can define $T_i=\|\,x_i-y_a\,\|_2^2$, as we are continuously increasing β.

Therefore, the equivalent energy function obtained by summing over all configurations of P is

$$-\frac{1}{\beta}\sum_{i=1}^{N}\log\sum_{a=1}^{M}\exp\{-\beta\|x_i-y_a\|_2^2\}+\frac{\kappa}{2}\sum_{a=1}^{M}\|y_a-y_{a\oplus 1}\|_2^2.$$

Explanation of Relevance of Objective Function(2)

The objective function given is:

$$E(P, P^{\text{old}}, Y) = \sum_{i=1}^{N}\sum_{a=1}^{M} P_{ia}\|x_i - y_a\|_2^2 + \frac{\kappa}{2}\sum_{a=1}^{M}\|y_a - y_{a\oplus 1}\|_2^2 + \frac{1}{\beta}\sum_{i=1}^{N}\sum_{a=1}^{M}\left(P_{ia}\log\frac{P_{ia}}{P_{ia}^{\text{old}}} - P_{ia} + P_{ia}^{\text{old}}\right)$$

Subject to the constraints $\sum_{a=1}^{M} P_{ia} = 1, \forall i \in \{1,\ldots,N\}$ and $P_{ia} > 0, \forall ia$.

We can define an elastic net in simple terms as

$$\hat{\beta} = \arg\min_{\beta} \| y - X\beta_3 \|_2^2 + \lambda_2 \| \beta_2 \|_2^2 + \lambda_1 \| \beta_1 \|_1$$

We can compare this equation with our objective function and can conclude the following:

$$y - X\beta_1 = \sum_{i=1}^{N}\sum_{a=a}^{M} P_{ia}(x_i - y_a), \beta_2 = \sum_{a=1}^{M} y_a - y_{a\oplus 1}, \lambda_2 = \frac{k}{2}, \lambda_1 = \frac{1}{\beta}, \beta_1 = \sum_{i=1}^{N}\sum_{a=1}^{M}(P_{ia}\log\frac{P_{ia}}{P_{ia}^{old}} - P_{ia} + P_{ia}^{old})$$

The part which contains the $\frac{1}{\beta}$ as λ_1 generates a sparse matrix.

The part which contains $\frac{k}{2}$ as λ_2 removes the limitation on the number of selected variables, encourages grouping effect and stabilizes the l1 regularization path.

Also, if we go deeper we can take the derivative of the objective function which will result as

$$d_a y_a - \sum_{i=1}^{N} P_{ia}x_i + k(2y_a - y_{a\oplus 1} - y_{a-\oplus 1}) + \frac{1}{\beta}\sum_{i=1}^{N}(P_{ia}^{old} + \log\frac{P_{ia}}{P_{ia}^{old}})$$

Here, $d_a \equiv \sum_{i=1}^{N} P_{ia}$ and $D_{aa} = d_a$.

Also, $L_{aa} = 2, L_{,a,a\oplus 1} = -1, L_{a,a-\oplus 1=-1}$.

Also,we know that the coordinate descent update for the j^{th} coefficient takes the form

$$\hat{\beta}_j = \frac{S_{\lambda\alpha}(\sum_{i=1}^{N} r_{ij}x_{ij})}{\sum_{i=1}^{N} x_{ij}^2 + \lambda(1-\alpha),}$$

Where $S_{\mu}(z) := sign(z)(z-\mu)_+$ is the soft-thresholding operator, and

$r_{ij} := y_i - \hat{\beta}_0 - \sum_{k\neq j} x_{ik}\hat{\beta}_k$ is the partial residual.

Here, we can solve this equation via $(kL + D)Y = P^T X$

Where $P_{ia}^{(new)} = \frac{P_{ia}^{(Old)} \exp\{-\beta \| x_i - y_a \|_2^2\}}{\sum_{b=1}^{M} P_{ib}^{(Old)} \exp\{-\beta \| x_i - y_b \|_2^2\}}$

Also,we have proved below the relevance of $y_a \oplus 1$ operation as representation invariant.

Let $y_a^{'} = g.y_a$ for $g \in H_t$ .We claim that $n_{y_a^{'}} = g n_{y_a} h^{-1}$.

The normalizing action is defined by inserting g^{-1} in parallel before both arguments in the last step.

$$n_{y_a} = \arg\min_{n' \in H_t} d(y_a^{'}, n^{'}) = \arg\min_{n' \in H_t} d(g y_a, n^{'} h) = \arg\min_{n' \in H_t} d(y_a, g^{-1} n^{'} h)$$

Since the normalizing action is unique,it follows that for the n' realizing the minimum it holds that $g^{-1} n' h = n_{y_a}$ and therefore $n' = n_{y_a^{'}} = g n_{y_a} h^{-1}$.

So,the sum $y_a^{'} \oplus 1 = y_a^{'} + n_{y_a^{'}} = g y_a + (g n_{y_a} h^{-1}) h = g.(y_a + n_{y_a}) \sim y_a \oplus 1$
Which proves the representation independence.

Thus, as we are using the $P_{ia}^{(new)} = \frac{P_{ia}^{(Old)} \exp\{-\beta \| x_i - y_a \|_2^2\}}{\sum_{b=1}^{M} P_{ib}^{(Old)} \exp\{-\beta \| x_i - y_b \|_2^2\}}$ formula in β_1 and since with every iteration λ_2 (which is 1/ β) is increasing, the elastic neuron matrix function also increases which forces the distance matrix objective function to convergence.

Thus,this leads to the solution of the global minima for the initial objective function given below.

$$E_{\mathrm{TSP}}(P, Y) = \sum_{i=1}^{N} \sum_{a=1}^{M} P_{ia} \| x_i - y_a \|_2^2 + \frac{\kappa}{2} \sum_{a=1}^{M} \| y_a - y_{a \oplus 1} \|_2^2.$$

Conclusion

We were able to implement the Elastic Net objective function in TSP100 as well as in TSP48 and were able to find a good approximation optimal tour length.Through,this project we were able to learned about the softmax non-linearity and also got some idea about recurrent neural networks,least squares and also how objective functions are coded in python.Since,Travelling Salesman Problem is a NP-Hard problem and finding an optimal length tour is an optimization problem,thus, we were also able to learn about convex optimization through this elastic net implementation which is a very useful topic in Machine Learning.