Data reduction and fitting

Eugeniy E. Mikhailov

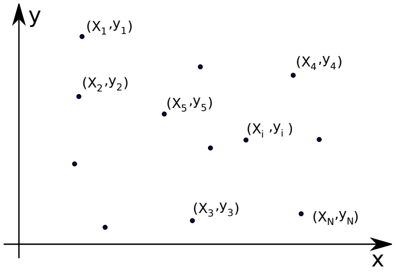
The College of William & Mary

Lecture 18

Traveling salesman problem

This problem has a lot of connection to the real life. Every time you ask your GPS to find a route, the GPS unit has to solve this problem. Layout of traces on a printed circuits board is essentially the same problem as well.

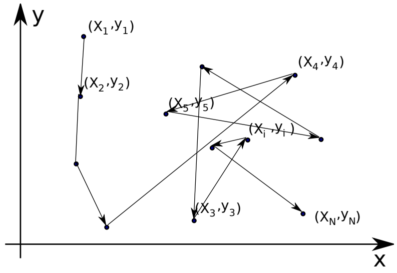
- Suppose you have N cities (with given coordinates) to visit
- Salesman start in the city 1 and need to be in the city *N* at the end of the route
- Find the shortest route so salesman visits every city only once



Traveling salesman problem

This problem has a lot of connection to the real life. Every time you ask your GPS to find a route, the GPS unit has to solve this problem. Layout of traces on a printed circuits board is essentially the same problem as well.

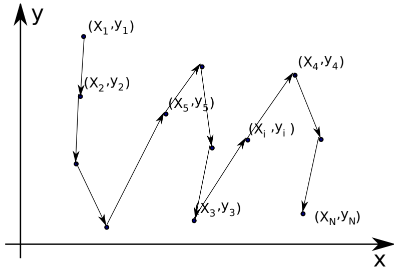
- Suppose you have N cities (with given coordinates) to visit
- Salesman start in the city 1 and need to be in the city *N* at the end of the route
- Find the shortest route so salesman visits every city only once



Traveling salesman problem

This problem has a lot of connection to the real life. Every time you ask your GPS to find a route, the GPS unit has to solve this problem. Layout of traces on a printed circuits board is essentially the same problem as well.

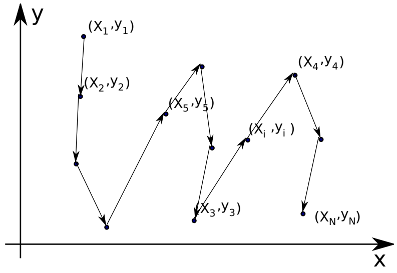
- Suppose you have N cities (with given coordinates) to visit
- Salesman start in the city 1 and need to be in the city N at the end of the route
- Find the shortest route so salesman visits every city only once



Traveling salesman problem

This problem has a lot of connection to the real life. Every time you ask your GPS to find a route, the GPS unit has to solve this problem. Layout of traces on a printed circuits board is essentially the same problem as well.

- Suppose you have N cities (with given coordinates) to visit
- Salesman start in the city 1 and need to be in the city N at the end of the route
- Find the shortest route so salesman visits every city only once



Note that combinatorial complexity of this problem

$$(N-2)!$$

since ends points are fixed. This grows very fast with N.

Possible solutions

- Brute force
  - Try every possible combination of cities and choose the best one
  - Will work for the modest route with $N \leq 10$ or may be slightly more

Possible solutions

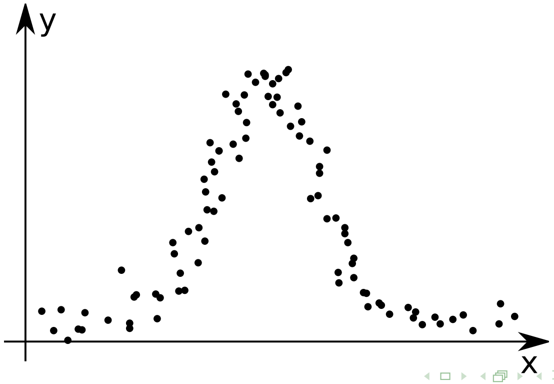
- Brute force
  - Try every possible combination of cities and choose the best one
  - Will work for the modest route with $N \leq 10$ or may be slightly more
- Metropolis algorithm
  - try routes probabilistically and anneal to the local optimum
    - This is reasonably fast
    - but does not guaranteed the best/shortest route
  - reasonable idea for generating a new route is to randomly swap 2 cities in the old route

Data reduction

- Typical modern experiment generates Mega bytes or even Terra bytes of data.

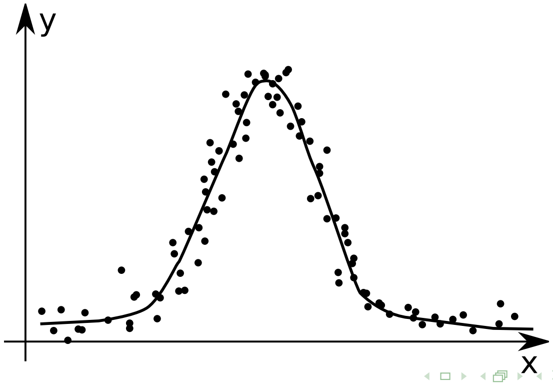
Data reduction

- Typical modern experiment generates Mega bytes or even Terra bytes of data.
- there is no way for a human to comprehend such enormous amount of data



Data reduction

- Typical modern experiment generates Mega bytes or even Terra bytes of data.
- there is no way for a human to comprehend such enormous amount of data
  - we need to post-process it and extract some important parameters
  - alternatively we want to check how our models reflect reality



Fitting

Someone measured bunch of experimental points y as a function of independent variable x. We want to extract model parameters $\vec{p}$ via fitting of the model function $f(x, \vec{p})$.

Remark: in general x and y could be vectors i.e. multi-dimensional, for example $\vec{x}$ has 2 coordinates: speed of the car and the weight of the load, and y would have the fuel consumption and the engine temperature.

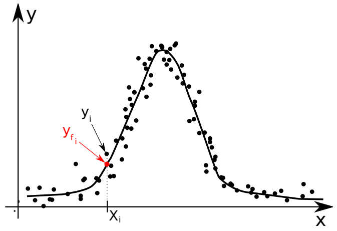
For simplicity we will focus on the one dimensional case for x and y

- we are given experimental points $x_i \rightarrow y_i$
- our model function $x_i \rightarrow y_{f_i} = f(x_i, \vec{p})$

Goodness of the fit

First we need to define some way to estimate goodness of the fit.

Very common is to use the sum of the squares of the fit deviations from the experiment data points.



$$\chi^2 = \sum_i (y_i - y_{f_i})^2$$

Differences of $(y_i - y_{f_i})$ are called residuals

For a given x, y and f the goodness of the fit χ^2 depends only on parameters of the model/fit function $\vec{p}$

So our job is simple, minimize χ^2 using any suitable algorithm. Thus find optimal $\vec{p}$. So called the least square fit.

Good fit should have the following properties

- residuals should be randomly scattered around 0
  - i.e. no visible trends of residuals vs x
- RMS residual $= \sqrt{\frac{1}{N}\sum_i^N (y_i - y_{f_i})^2}$ should be in order of the Δy (experimental uncertainty for y)

For practical realization of the fitting algorithm have a look at the 'fitter.m' file posted at the class web page. See also 'fitter_usage_example.m'