Simulated annealing/Metropolis optimization

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 17

Example: Backpack problem

Suppose you have a backpack which has a given size (volume). You have a set of objects with given volumes and values (for example their cost).

Our job is to find a such subset of items that still fits in the backpack and has the maximum combined value.

In simple case we will assume that every items happen only once. Then our job is to maximize

$$E(\vec{x}) = \sum value_i x_i$$

Subject of the following constrains

$$\sum volume_i x_i \leq \mathrm{BackpackSize}$$

Where $x_i = (0 \text{ or } 1)$ i.e. do we take this object or not

Brute force combinatorial backpack optimization

Recall that we are looking for an optimal direction among all possible $\vec{x}$
Generally $\vec{x}$ is combo of zeros and ones $\vec{x} = [0, 1, 0, 1, \cdots, 1, 1, 0, 1, 1]$
How would we generate all possible combinations of $\vec{x}$ components?

Brute force combinatorial backpack optimization

Recall that we are looking for an optimal direction among all possible $\vec{x}$
Generally $\vec{x}$ is combo of zeros and ones $\vec{x} = [0, 1, 0, 1, \cdots, 1, 1, 0, 1, 1]$
How would we generate all possible combinations of $\vec{x}$ components?

- $\vec{x}$ looks like binary number.
- let's start with $\vec{x} = [0, 0, 0, 0, \cdots, 0, 0]$
- every new components will be generated by adding 1 to the previous x according to binary addition rules
  - for example
    $x_{next} = [1, 0, 1, \cdots, 1, 1, 0, 1, 1] + 1 = [1, 0, 1, \cdots, 1, 1, 1, 0, 0]$
- for every new $\vec{x}$ we check if items fit to backpack and if new fitted value is larger then previous
- once we tried all 2^N combinations of $\vec{x}$ we are done

- So time of optimization grows exponentially with the number N of items to chose.
- But we will find the global optimum.

Brute force combinatorial backpack optimization

For realization of this algorithm have a look at the 'backpack_binary.m' file.

- My computer sorts 20 items in 47 seconds, but 30 items would take 1000 times longer something like 13 hours to solve.

Nature's way ti find a minimum energy

- We see that probing full space permitted by combinatorics is not practical even for a reasonably small size problem.

Nature's way ti find a minimum energy

- We see that probing full space permitted by combinatorics is not practical even for a reasonably small size problem.
- However nature seems to handle the problem of the energy minimization without any trouble.

Nature's way ti find a minimum energy

- We see that probing full space permitted by combinatorics is not practical even for a reasonably small size problem.
- However nature seems to handle the problem of the energy minimization without any trouble.
- For example, if you heat up a piece of metal and then slowly cool it i.e. anneal, then the system will reach the minimum energy state.

Simulated annealing/Metropolis algorithm

Metropolis and coworker suggested in 1953 the following heuristic algorithm based on this observation, and the Boltzmann energy distribution law.

1. set the temperature to a high value so kT is larger then typical energy (merit) function fluctuation.
   - This requires some experiments if you do not know this a priori.
2. assign the $\vec{x}$ and calculate the energy at this point E.
3. change somehow old $\vec{x}$ to generate a new one $\vec{x}_{new}$
4. calculate the energy at new point $E_{new} = E(\vec{x})$
5. if $E_{new} < E$ then $x = x_{new}$ and $E = E_{new}$
   - i.e. we move to new point of the lower energy
6. otherwise move to the new point with probability $p = exp(-(E_{new} - E)/kT)$
   - this resembles the Boltzmann energy distribution probability
7. decrease the temperature a bit i.e. keep annealing
8. repeat from the step 3 for a given number of cycles
9. $\vec{x}$ will hold the local optimal solution

Simulated annealing/Metropolis algorithm facts

In finite time (limited number of cycles) the algorithm guaranteed to find only local minimum.

Simulated annealing/Metropolis algorithm facts

In finite time (limited number of cycles) the algorithm guaranteed to find only local minimum.
There is a theorem which states:

The probability to find the best solution goes to 1, as we run algorithm for a longer time with a slow rate of cooling.

Simulated annealing/Metropolis algorithm facts

In finite time (limited number of cycles) the algorithm guaranteed to find only local minimum.
There is a theorem which states:

> The probability to find the best solution goes to 1, as we run algorithm for a longer time with a slow rate of cooling.

Unfortunately, this theorem is of no use since it does not give a recipe of how long to run the algorithm. It is even suggested that it will need more cycles than the brute force combinatorial search.

Simulated annealing/Metropolis algorithm facts

In finite time (limited number of cycles) the algorithm guaranteed to find only local minimum.
There is a theorem which states:

> The probability to find the best solution goes to 1, as we run algorithm for a longer time with a slow rate of cooling.

Unfortunately, this theorem is of no use since it does not give a recipe of how long to run the algorithm. It is even suggested that it will need more cycles than the brute force combinatorial search.
However, in practice very good solutions can be found in quite short time with quite small number of cycles.

Simulated annealing/Metropolis algorithm facts

In finite time (limited number of cycles) the algorithm guaranteed to find only local minimum.
There is a theorem which states:

> The probability to find the best solution goes to 1, as we run algorithm for a longer time with a slow rate of cooling.

Unfortunately, this theorem is of no use since it does not give a recipe of how long to run the algorithm. It is even suggested that it will need more cycles than the brute force combinatorial search.
However, in practice very good solutions can be found in quite short time with quite small number of cycles.
The Metropolis algorithm method is not limited to the discrete space problems, and can be used for the problems accepting real values of the $\vec{x}$ components.

Simulated annealing/Metropolis algorithm facts

In finite time (limited number of cycles) the algorithm guaranteed to find only local minimum.
There is a theorem which states:

> The probability to find the best solution goes to 1, as we run algorithm for a longer time with a slow rate of cooling.

Unfortunately, this theorem is of no use since it does not give a recipe of how long to run the algorithm. It is even suggested that it will need more cycles than the brute force combinatorial search.
However, in practice very good solutions can be found in quite short time with quite small number of cycles.
The Metropolis algorithm method is not limited to the discrete space problems, and can be used for the problems accepting real values of the $\vec{x}$ components.

- the main challenge is to find a good way to choose new $\vec{x}$ to probe.

Backpack problem with Metropolis algorithm

- The main challenge is to find a good routine to generate new candidate for the $\vec{x}_{new}$.
- Recall that $\vec{x}$ generally looks like $[0, 1, 1, 0, 1, \cdots, 0, 1, 1]$ so lets just randomly toggle, i.e. mutate, the choices with some probability.
- The rest is quite straight forward, as long as we remember, that we are looking for the maximum value in the backpack, while Metropolis algorithm is designed for merit function minimization. So we choose our merit function to be negative value of all items in the backpack. Also we need to add a big penalty for the case of the overfilled backpack.
- See the realization of the algorithm in the 'backpack_metropolis.m' file.
- it will find quite good solution for the 30 items to choose problem within a second instead of 13 hours of combinatorial search.