Submodular function maximization and Greedy algorithm

yataka

2018 年 12 月 10 日

1　はじめに

この方程では V は有限集合であるとする.

2　劣モジュラ関数

まず劣モジュラ関数を定義する.

Definition 2.1. (Discreate derivative)
f を集合関数とする. S を V の部分集合とし e を V の元とする. この時 f の S での Discreate derivative を

$$\Delta_f(e|S) := f(S \cup \{e\}) - f(S)$$

と定義する.

Definition 2.2. (Submodular function)
f を集合関数とする. f が Submodular function であるとは

$$\forall A, B \subset V, \forall e \in V, A \subset B \text{ かつ } e \in V - B \Longrightarrow \Delta_f(e|A) \leq \Delta_f(e|B)$$

を満たすことをいう.

Theorem 2.3. f を集合関数とする. この時

1. f は Submodular function である.
2. $\forall A, B \subset V, f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$

は同値である.

まあ, ここまでよくわからない Submodular function というものを定義して来たが, 読者の中には「V は有限集合なんだから, V の要素を入っているか入っていないかで分けて全通り試せば良い.」と思う人もいるかもしれない. 実際にそれを C++ で実装すると以下のようになる.

Listing 1　C++ のソースコード

```cpp
#include<iostream>
#include<math.h>
using namespace std;
#define MAX_N 10
int main(){
    int f_V[MAX_N];
    for(int i = 0;i < MAX_N; i++){
        if(i % 3 ==0){
            f_V[i] = -2 * i + 1;
        }else{
            f_V[i] = 2 * i + 1;
        }
    }
    int sum[1024] = {0};
    for(int i = 0; i < pow(2, MAX_N); i++){
        for(int k = 0; k < MAX_N; k++){
            if(i & 1<<k){
                sum[i] += f_V[k];
            }
        }
    }
    for(int i = 0; i < 1024; i++){
        cout << sum[i] << " ";
    }
    cout << endl;
    return 0;
}
```

このコードでは, $|V| = 10$ の場合を考えて実装している．このように $|V|$ が小さい場合であればこのような方法でも問題はない．しかしながら, $|V| = 10000$ の場合はどうだろうか? 調べる候補の数は 2^{10} から 2^{10000} へと跳ね上がる．この量を全て調べ上げるのは現実的に不可能である.(100 年とか待てるなら別ですが...)

というわけで, 集合関数 f が最大となる V の部分集合 S を効率に求めることが必要なのである.