STA 5106
Computational Methods in Statistics I

Department of Statistics
Florida State University

Class 22
November 19, 2019

Special Topic 2

Dynamic Programming

Dynamic Programming

- Dynamic Programming has emerged as an important tool in finding optimal paths in a variety of situations.
- This tool is useful in finding an optimal trajectory in situations where the decisions are made in stages and decisions made in past stages have future consequences.
- In particular, the decisions are made at discrete indices.
- The cost associated with a trajectory is the sum of costs associated with these individual decisions.

Example One

- Assume we want to compute the following sum

$$S = \sum_{x_1=1}^{K} \sum_{x_2=1}^{K} \cdots \sum_{x_N=1}^{K} (x_1 x_2 + x_2 x_3 + \ldots + x_{N-1} x_N)$$

- **Brute-force computation:** the number of computational steps is in the order of NK^N. That is, there exists a constant C, such that

$$\frac{\text{Number of computational steps}}{NK^N} \leq C$$

for all N.

- We write it as $O(NK^N)$.

Graphical Model

$$x_1 \;—\; x_2 \;—\; x_3 \;—\; \cdots\cdots \;—\; x_N$$

- Each x_i only depends on its neighbors $x_{i\text{-}1}$ and x_{i+1}. Therefore,

$$S = \sum_{x_1=1}^{K}[\sum_{x_2=1}^{K}[K^{N-2}x_1x_2 + \sum_{x_3=1}^{K}[K^{N-3}x_2x_3 + \cdots \sum_{x_{N-1}=1}^{K}[Kx_{N-2}x_{N-1} + \sum_{x_N=1}^{K}x_{N-1}x_N]...]]]$$

- Compute the sum in backward:

  1. For a given $x_{N\text{-}1}$, compute the sum on x_N.
  2. For a given $x_{N\text{-}2}$, compute the sum on $x_{N\text{-}1}$, where the sum on x_N is also included.
  3. Continue the process until the sum on x_1 is done.

Dynamic Programming

- Formally, we have the following recursive procedure to compute

$$S=\sum_{x_1=1}^{K}[\sum_{x_2=1}^{K}[K^{N-2}x_1x_2+\sum_{x_3=1}^{K}[K^{N-3}x_2x_3+\cdots\sum_{x_{N-1}=1}^{K}[Kx_{N-2}x_{N-1}+\sum_{x_N=1}^{K}x_{N-1}x_N]\ldots]]]$$

1. For each $x_{N\text{-}1}$, compute $h(x_{N-1})=\sum_{x_N=1}^{K}x_{N-1}x_N$

2. For each $x_{N\text{-}2}$, compute $h(x_{N-2})=\sum_{x_{N-1}=1}^{K}Kx_{N-2}x_{N-1}+h(x_{N-1})$

3. In general, compute $h(x_i)=\sum_{x_{i+1}=1}^{K}K^{N-i-1}x_ix_{i+1}+h(x_{i+1}),\quad i=N-2,N-1,\ldots,2$

4. Finally, $S=\sum_{x_1=1}^{K}h(x_1)$

- The computational order is $O(NK^2)$

Extension

- 1. The interaction can be any function.

General form:

$$S = \sum_{x_1=1}^{K}\sum_{x_2=1}^{K}\cdots\sum_{x_N=1}^{K}\left(f_1(x_1, x_2) + f_2(x_2, x_3) + \ldots + f_{N-1}(x_{N-1}, x_N)\right)$$

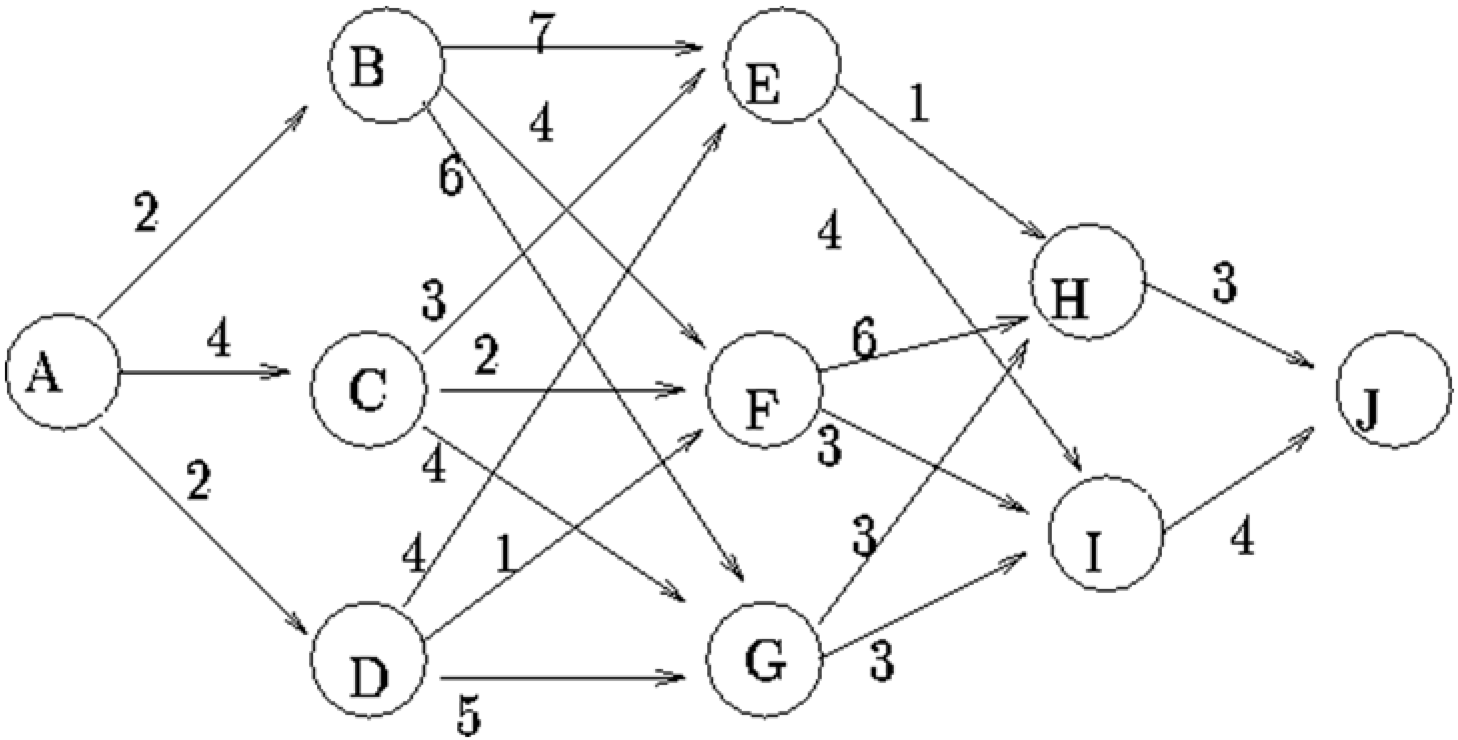
- 2. The interaction can be defined for more than two variables.

For example:

$$S = \sum_{x_1=1}^{K}\sum_{x_2=1}^{K}\cdots\sum_{x_N=1}^{K}\left(f_1(x_1, x_2, x_3) + f_2(x_3, x_4) + f_3(x_4, x_5, x_6) + \ldots + f_{N-1}(x_{N-1}, x_N)\right)$$

Example Two

- **Shortest path problem:** Find the shortest path from A to J in the following road network.



Brute-Force Computation

- ABEHJ: 2+7+1+3 = 13
- ABEIJ: 2+7+4+4 = 17
- ABFHJ: 2+4+6+3 = 15
- ABFIJ: 2+4+3+4 = 13
- ABGHJ: 2+6+3+3 = 14
- ABGIJ: 2+ 6+3+4 = 15
- …
- …
- ADGIJ: 2+5+3+4 = 14

- **Number of paths = 3x3x2 = 18**

Backward Dynamic Programming

- Let A be the 1st layer, J be the 5th layer, and let a, b, c denote the choice in the 2nd, 3rd, and 4th layers.

Backward dynamic programming:

- If b = E, then c = H $(4 < 8)$
- If b = F, then c = I $(7 < 9)$
- If b = G, then c = H $(6 < 7)$
- If a = B, then b = E or F $(11 = 11 < 12)$
- If a = C, then b = E $(7 < 9 < 10)$
- If a = D, then b = E or F $(8 = 8 < 11)$
- Therefore, a = D, b = E or F, c = H or I $(10 < 11 < 13)$; that is, the shortest path is ADEHJ or ADFIJ.

Forward Dynamic Programming

Forward dynamic programming:

- If b = E, then a = D (6 < 7 < 9)
- If b = F, then a = D (3 < 6 = 6)
- If b = G, then a = D (7 < 8 = 8)
- If c = H, then b = E (7 < 9 < 10)
- If c = I, then b = F (6 < 10 = 10)

- Therefore, a = D, b = E or F, c = H or I (10 =10); that is, the shortest path is ADEHJ or ADFIJ.

Extension

- The same dynamic programming procedure can be used for maximization/minimization and sum.

- For example, we can efficiently solve optimization problems such as:

$$S = \arg\max_{x_1, x_2, \ldots, x_N} (f_1(x_1, x_2) + f_2(x_2, x_3) + \ldots + f_{N-1}(x_{N-1}, x_N))$$

and

$$S = \arg\min_{x_1, x_2, \ldots, x_N} (f_1(x_1, x_2, x_3) + f_2(x_3, x_4) + f_3(x_4, x_5, x_6) + \ldots + f_{N-1}(x_{N-1}, x_N))$$