CSE 421

Bellman Ford – Linear Programming

Shayan Oveis Gharan
Shortest Paths with Negative Edge Weights
Shortest Paths with Neg Edge Weights

Given a weighted directed graph $G = (V, E)$ and a source vertex $s$, where the weight of edge $(u,v)$ is $c_{u,v}$

**Goal:** Find the shortest path from $s$ to all vertices of $G$.

Recall that Dijkstra’s Algorithm fails when weights are negative.
Observation: No solution exists if G has a negative cycle.

This is because we can minimize the length by going over the cycle again and again.

So, suppose G does not have a negative cycle.
DP for Shortest Path

Def: Let \( OPT(v, i) \) be the length of the shortest \( s - v \) path with at most \( i \) edges.

Let us characterize \( OPT(v, i) \).

Case 1: \( OPT(v, i) \) path has less than \( i \) edges.
- Then, \( OPT(v, i) = OPT(v, i - 1) \).

Case 2: \( OPT(v, i) \) path has exactly \( i \) edges.
- Let \( s, v_1, v_2, \ldots, v_{i-1}, v \) be the \( OPT(v, i) \) path with \( i \) edges.
- Then, \( s, v_1, \ldots, v_{i-1} \) must be the shortest \( s - v_{i-1} \) path with at most \( i - 1 \) edges. So,
  \[
  OPT(v, i) = OPT(v_{i-1}, i - 1) + c_{v_{i-1}, v}
  \]
DP for Shortest Path

Def: Let $OPT(v, i)$ be the length of the shortest $s$ - $v$ path with at most $i$ edges.

$$OPT(v, i) = \begin{cases} 
0 & \text{if } v = s \\
\infty & \text{if } v \neq s, i = 0 \\
\min(OPT(v, i - 1), \min_{u:(u,v) \text{ an edge}} OPT(u, i - 1) + c_{u,v}) & \text{otherwise}
\end{cases}$$

So, for every $v$, $OPT(v, \cdot)$ is the shortest path from $s$ to $v$. But how long do we have to run? Since $G$ has no negative cycle, it has at most $n - 1$ edges. So, $OPT(v, n - 1)$ is the answer.
Bellman Ford Algorithm

for v=1 to n
    if \( v \neq s \) then
        \( M[v,0]=\infty \)
    \( M[s,0]=0. \)

for i=1 to n-1
    for v=1 to n
        \( M[v,i]=M[v,i-1] \)
        for every edge \((u,v)\)
            \( M[v,i]=\min(M[v,i], M[u,i-1]+c_{u,v}) \)

Running Time: \( O(nm) \)
Can we test if G has negative cycles?
for $v=1$ to $n$
  if $v \neq s$ then
    $M[v,0]=\infty$
  $M[s,0]=0.$

for $i=1$ to $n-1$
  for $v=1$ to $n$
    $M[v,i]=M[v,i-1]$
    for every edge $(u,v)$
      $M[v,i]=\min(M[v,i], M[u,i-1]+c_{u,v})$

**Running Time:** $O(nm)$
Can we test if G has negative cycles? Yes, run for $i=1\ldots2n$ and see if the $M[v,n-1]$ is different from $M[v,2n]$
System of Linear Equations

Find a solution to

\[\begin{align*}
x_3 - x_1 &= 4 \\
x_3 - 2x_2 &= 3 \\
x + 2x_2 + x_3 &= 7
\end{align*}\]

Can be solved by Gaussian elimination method
Linear Programming

Optimize a linear function subject to linear inequalities

\[
\begin{align*}
\text{max} & \quad 3x_1 + 4x_3 \\
\text{s.t.} & \quad x_1 + x_2 \leq 5 \\
& \quad x_3 - x_1 = 4 \\
& \quad x_3 - x_2 \geq -5 \\
& \quad x_1, x_2, x_3 \geq 0
\end{align*}
\]

• We can have inequalities,
• We can have a linear objective functions
Applications of Linear Programming

Generalizes: $Ax=b$, 2-person zero-sum games, shortest path, max-flow, matching, multicommodity flow, MST, min weighted arborescence, …

Why significant?
• We can solve linear programming in polynomial time.
• Useful for approximation algorithms
• We can model many practical problems with a linear model and solve it with linear programming

Linear Programming in Practice:
• There are very fast implementations: IBM CPLEX, Gorubi in Python, CVX in Matlab, ….
• CPLEX can solve LPs with millions of variables/constraints in minutes
Example 1: Diet Problem

Suppose you want to schedule a diet for yourself. There are four category of food: veggies, meat, fruits, and dairy. Each category has its own (p)rice, (c)alory and (h)appiness per pound:

<table>
<thead>
<tr>
<th></th>
<th>veggies</th>
<th>meat</th>
<th>fruits</th>
<th>dairy</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>$p_v$</td>
<td>$p_m$</td>
<td>$p_f$</td>
<td>$p_d$</td>
</tr>
<tr>
<td>calorie</td>
<td>$c_v$</td>
<td>$c_m$</td>
<td>$c_f$</td>
<td>$c_d$</td>
</tr>
<tr>
<td>happiness</td>
<td>$h_v$</td>
<td>$h_m$</td>
<td>$h_f$</td>
<td>$h_d$</td>
</tr>
</tbody>
</table>

**Linear Modeling:** Consider a linear model: If we eat 0.5lb of meat, 0.2lb of fruits we will be $0.5 \times h_m + 0.2 \times h_f$ happy

- You should eat 1500 calories to be healthy
- You can spend 20 dollars a day on food.

**Goal:** Maximize happiness?
Diet Problem by LP

- You should eat 1500 calories to be healthy
- You can spend 20 dollars a day on food.

Goal: Maximize happiness?

<table>
<thead>
<tr>
<th></th>
<th>veggies</th>
<th>meat</th>
<th>fruits</th>
<th>dairy</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>( p_v )</td>
<td>( p_m )</td>
<td>( p_f )</td>
<td>( p_d )</td>
</tr>
<tr>
<td>calorie</td>
<td>( c_v )</td>
<td>( c_m )</td>
<td>( c_f )</td>
<td>( c_d )</td>
</tr>
<tr>
<td>happiness</td>
<td>( h_v )</td>
<td>( h_m )</td>
<td>( h_f )</td>
<td>( h_d )</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{max} & \quad x_v h_v + x_m h_m + x_f h_f + x_d h_d \\
\text{s.t.} & \quad x_v p_v + x_m p_m + x_f p_f + x_d p_d \leq 20 \\
& \quad x_v c_v + x_m c_m + x_f c_f + x_d c_d \leq 1500 \\
& \quad x_v, x_m, x_f, x_d \geq 0
\end{align*}
\]

#pounds of veggies, meat, fruits, dairy to eat per day
How to Design an LP?

• Define the set of variables

• Put constraints on your variables,
  • should they be nonnegative?

• Write down the constraints
  • If a constraint is not linear try to approximate it with a linear constraint

• Write down the objective function
  • If it is not linear approximation with a linear function

• Decide if it is a minimize/maximization problem
Example 2: Max Flow

Define the set of variables
• For every edge $e$ let $x_e$ be the flow on the edge $e$

Put constraints on your variables
• $x_e \geq 0$ for all edge $e$ (The flow is nonnegative)

Write down the constraints
• $x_e \leq c(e)$ for every edge $e$, (Capacity constraints)
• $\sum_{e \text{ out of } v} x_e = \sum_{e \text{ in to } v} x_e \ \forall v \neq s, t$ (Conservation constraints)

Write down the objective function
• $\sum_{e \text{ out of } s} x_e$

Decide if it is a minimize/maximization problem
• max
Example 2: Max Flow

\[
\begin{align*}
\text{max} & \quad \sum_{e \text{ out of } s} x_e \\
\text{s.t.} & \quad \sum_{e \text{ out of } v} x_e = \sum_{e \text{ in to } v} x_e & \forall v \neq s, t \\
& \quad x_e \leq c(e) & \forall e \\
& \quad x_e \geq 0 & \forall e
\end{align*}
\]

Q: Do we get exactly the same properties as Ford Fulkerson?
A: Not necessarily, the max-flow may not be integral
Example 3: Min Cost Max Flow

Suppose we can route 100 gallons of water from $s$ to $t$. But for every pipe edge $e$ we have to pay $p(e)$ for each gallon of water that we send through $e$.

**Goal:** Send 100 gallons of water from $s$ to $t$ with minimum possible cost

$$\min \sum_{e \in E} p(e) \cdot x_e$$
$$s.t. \sum_{e \text{ out of } v} x_e = \sum_{e \text{ into } v} x_e, \forall v \neq s, t$$
$$\sum_{e \text{ out of } s} x_e = 100$$
$$x_e \leq c(e), \forall e$$
$$x_e \geq 0, \forall e$$
Summary (Linear Programming)

- Linear programming is one of the biggest advances in 20th century

- It is being used in many areas of science: Mechanics, Physics, Operations Research, and in CS: AI, Machine Learning, Theory, …

- Almost all problems that we talked can be solved with LPs, Why not use LPs?
  - Combinatorial algorithms are typically faster
  - They exhibit a better understanding of worst case instances of a problem
  - They give certain structural properties, e.g., Integrality of Max-flow when capacities are integral

- There is rich theory of LP-duality which generalizes max-flow min-cut theorem
What is next?

• CSE 431 (Complexity Course)
  • How to prove lower bounds on algorithms?

• CSE 521 (Graduate Algorithms Course)
  • How to design streaming algorithms?
  • How to design algorithms for high dimensional data?
  • How to use matrices/eigenvalues/eigenvectors to design algorithms
  • How to use LPs to design algorithms?

• CSE 525 (Graduate Randomized Algorithms Course)
  • How to use randomization to design algorithms?
  • How to use Markov Chains to design algorithms?