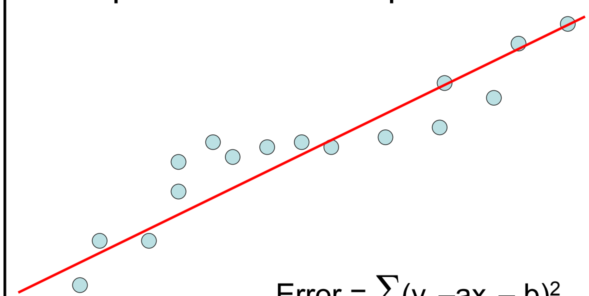
CSE 421
Algorithms

Richard Anderson

Lecture 17

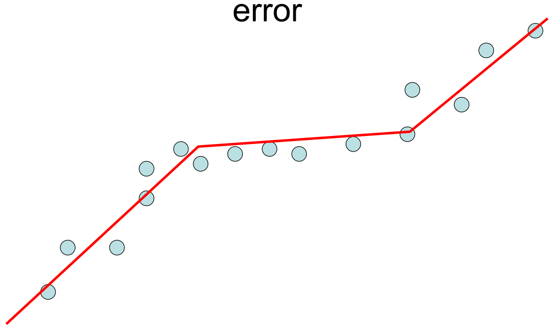
Dynamic Programming

Optimal linear interpolation



Error = $\sum(y_i - ax_i - b)^2$

Determine set of K lines to minimize error



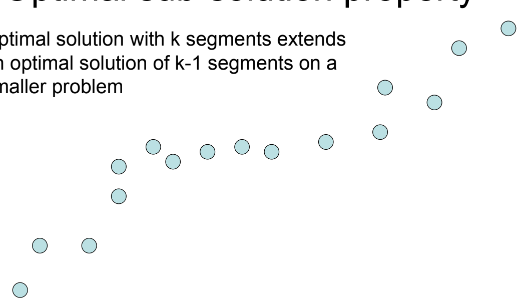
$Opt_k[\ j\ ]$: Minimum error approximating $p_1 \ldots p_j$ with k segments

Express $Opt_k[\ j\ ]$ in terms of $Opt_{k-1}[1], \ldots, Opt_{k-1}[\ j\ ]$

$Opt_k[\ j\ ] = \min_i \{Opt_{k-1}[\ i\ ] + E_{i,j}\}$

Optimal sub-solution property

Optimal solution with k segments extends an optimal solution of k-1 segments on a smaller problem



Optimal multi-segment interpolation

Compute Opt[k, j] for 0 < k < j < n

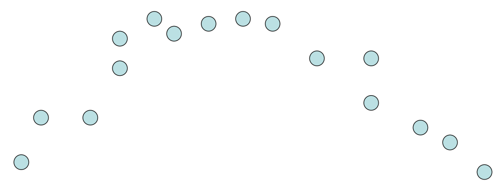
```
for j := 1 to n
   Opt[ 1, j] = E_{1,j};
for k := 2 to n-1
   for j := 2 to n
       t := E_{1,j}
       for i := 1 to j -1
          t = min (t, Opt[k-1, i ] + E_{i,j})
       Opt[k, j] = t
```

Determining the solution

- When Opt[k ,j] is computed, record the value of i that minimized the sum
- Store this value in a auxiliary array
- Use to reconstruct solution

Variable number of segments

- Segments not specified in advance
- Penalty function associated with segments
- Cost = Interpolation error + C x #Segments



Penalty cost measure

- $Opt[j] = \min(E_{1,j}, \min_i(Opt[i] + E_{i,j})) + P$

Subset Sum Problem

- Let $w_1, \ldots, w_n$ = {6, 8, 9, 11, 13, 16, 18, 24}
- Find a subset that has as large a sum as possible, without exceeding 50

Adding a variable for Weight

- Opt[j, K] the largest subset of $\{w_1, \ldots, w_j\}$ that sums to at most K
- {2, 4, 7, 10}
  - Opt[2, 7] =
  - Opt[3, 7] =
  - Opt[3,12] =
  - Opt[4,12] =

Subset Sum Recurrence

- Opt[j, K] the largest subset of $\{w_1, \ldots, w_j\}$ that sums to at most K

Subset Sum Grid

Opt[j, K] = max(Opt[j – 1, K], Opt[j – 1, K – w_j] + w_j)

4																	
3																	
2																	
1																	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

{2, 4, 7, 10}

Subset Sum Code

Knapsack Problem

- Items have weights and values
- The problem is to maximize total value subject to a bound on weght
- Items {I_1, I_2, … I_n}
  - Weights {w_1, w_2, …,w_n}
  - Values {v_1, v_2, …, v_n}
  - Bound K
- Find set S of indices to:
  - Maximize $\sum_{i \varepsilon S} v_i$ such that $\sum_{i \varepsilon S} w_i$ <= K

Knapsack Recurrence

Subset Sum Recurrence:

Opt[j, K] = max(Opt[j – 1, K], Opt[j – 1, K – w_j] + w_j)

Knapsack Recurrence:

Knapsack Grid

Opt[j, K] = max(Opt[j – 1, K], Opt[j – 1, K – w_j] + v_j)

4																	
3																	
2																	
1																	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Weights {2, 4, 7, 10} Values: {3, 5, 9, 16}