innovate achieve lead

BITS Pilani

WILP

AMLCCLZG516
ML System Optimization

Murali Parameswaran

innovate achieve lead

BITS Pilani

WILP

AML CCLZG516
ML System Optimization

Parallel/Distributed ML Algorithms

- Exercises: kNN, Decision Trees
- Assignment

Nearest Neighbors method for Classification

- Approach:
  - A (test) point **p** is assigned to a class **C**, to which belong a majority of the neighbors of **p**
    - This requires determining the nearest neighbors
  - The number of neighbors, K,
    - to be used to identify the class of a given test point is chosen externally
    - and the choice is non-trivial:
      - Too Large K ==> reduced accuracy
      - Too Small K ==> increased noise-sensitivity

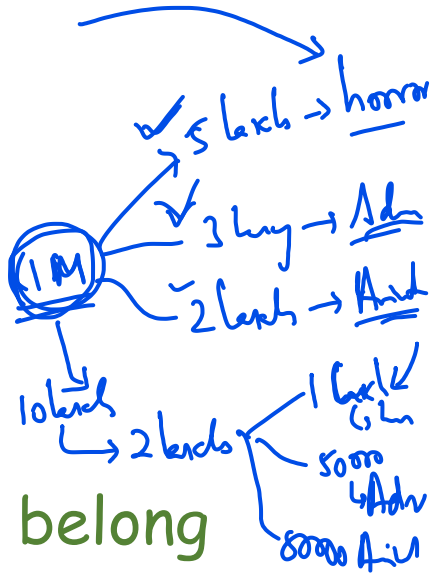
K Nearest Neighbors: Algorithm

Inputs: Dataset D, positive integer k // D is (partially) labelled

Approach:

1. for each x in D:
   1. Compute distances to all other points in D
   2. Choose the k neighboring points nearest to x
   3. Identify the class C to which a majority of these neighbors belong
   4. x.class = C

Break ties arbitrarily!



K Nearest Neighbors: Algorithm: Analysis

Inputs: Dataset D, positive integer k // |D|=N

Approach:

1. for each x in D:
    1. Compute distances to all other points in D
    2. Choose the k neighboring points nearest to x
    3. Identify the class C to which a majority of these neighbors belong
    4. x.class = C

- $O(N^2)$ distance computations
- Each computation take O(d) time
    - (assuming d-dimensional Euclidean space).

What about step 2?
Sorting of distances:
N. Log(N)
Extract k shortest values

- O(N) majority computations
- Each computation takes O(k) time

- Loop is running for each point: O(N) times

- Step 1: O(d).O(N) // $O(d. N^2)$
- Step 2: O(N. log N) // $O(N^2 \log N)$
- Step 3: O(k) // O(k. N)

K Nearest Neighbors: Algorithm: Analysis

Inputs: Dataset D, positive integer k

Approach:

1. for each x in D:
    1. Compute distances to all other points in D
    2. Choose the k neighboring points nearest to x
    3. Identify the class C to which a majority of these neighbors belong
    4. x.class = C

Step 2:

- Naïve approach: Sort the neighbors by distance! // O(NlogN) additional time

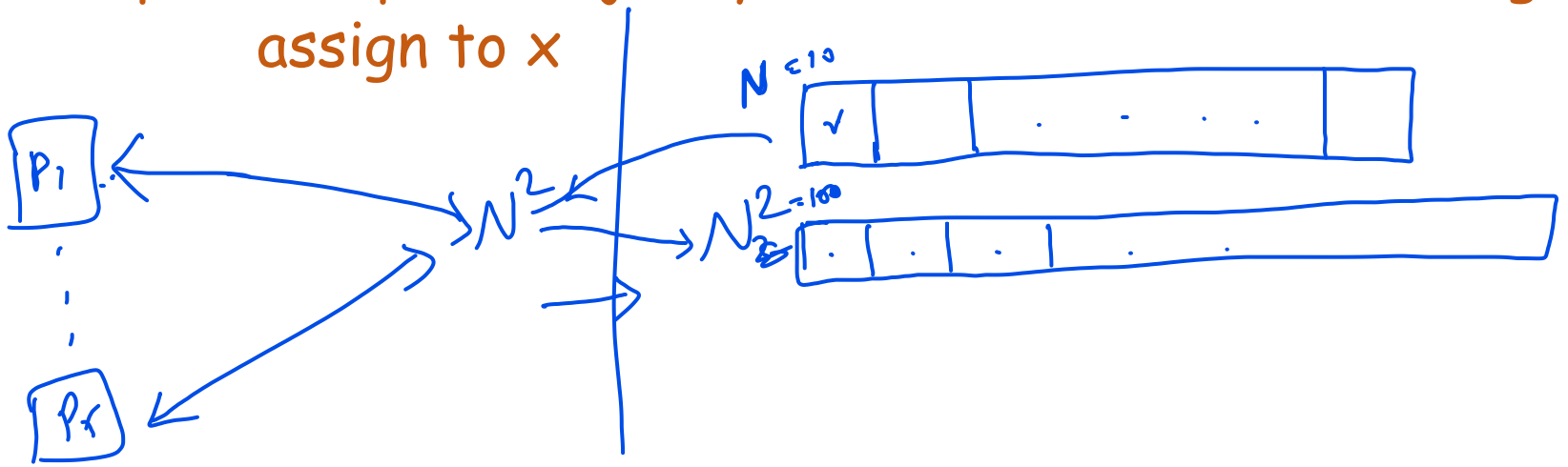
A popular alternative: Construct a KD-tree (using the distance metric)

- Combined cost for distance computations and tree construction:
    - O(d*N*logN), where for d-dimensional data

Shared Memory processor

Parallel Algorithm - Data parallel

- How do you parallelize kNN?
  - for each x in D:
  - Step 1: Distance Computation
    - Distance from a point x to every other point.
  - Step 2: Getting the nearest neighbors
  - Step 3: Compute majority Class of the k nearest neighbours, and assign to x



Shared Memory processor

Parallel Algorithm - Data parallel

- How do you parallelize kNN?
  - for each x in D:
  - Step 1: Distance Computation
    - N^2 processors: 1 distance computation each;
    - p processors: N^2/p computation in each processor
    - ==========
  - Step 2: Getting the nearest neighbors
  - Option 1: for each x in D:
    - Parallel-sort the neighbors by distance (and select the first k)
  - Option 2: for each processor P_j j = 1 to p
  - for each x in D_j
    - sort the neighbors (all points) by distance // sorting of |D| values
    - ==========
  - Step 3: for each processor P_j j = 1 to p
  - for each x in D_j
    - Finding the majority among k neighbors

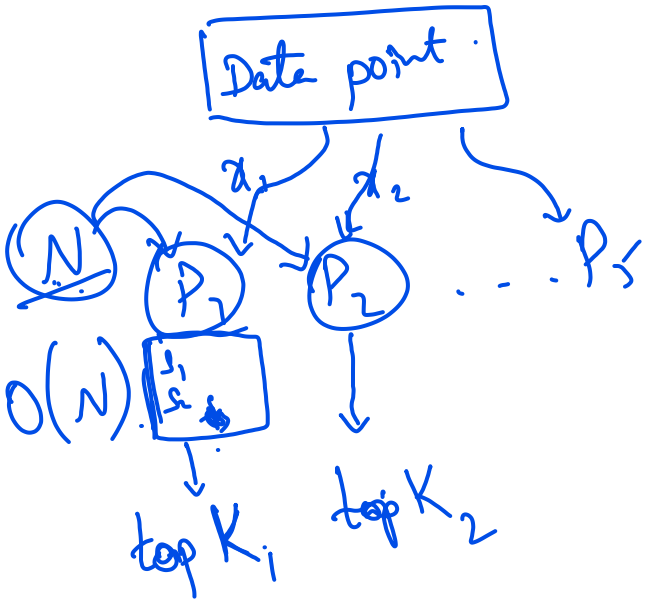
Speedup: p

Speedup: p

Speedup: p

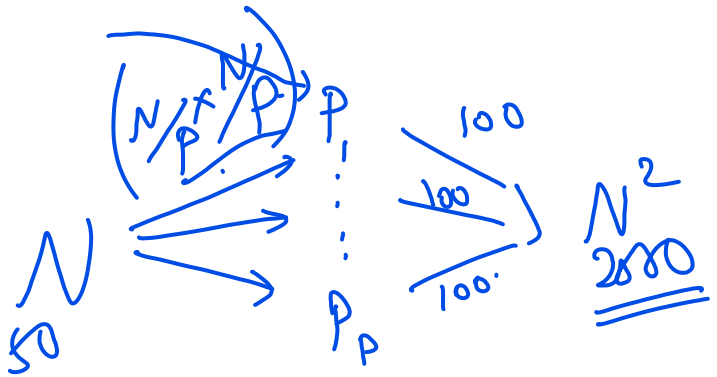
Online Algorithm – Request Parallelism

- Maintain a pool of threads
  - When a point p arrives, it is assigned to a (free) thread
- (thread == processor)



IF Data points are coming in a stream, one after another?

kNN: Distributed Algorithm



- How do you distribute kNN ?
  - Step 1: Distance Computation
  - Step 2: Getting the nearest neighbors
  - Step 3: Finding the majority

    - E.g. Step 1: point 1 (at node) – distance computation with all other points

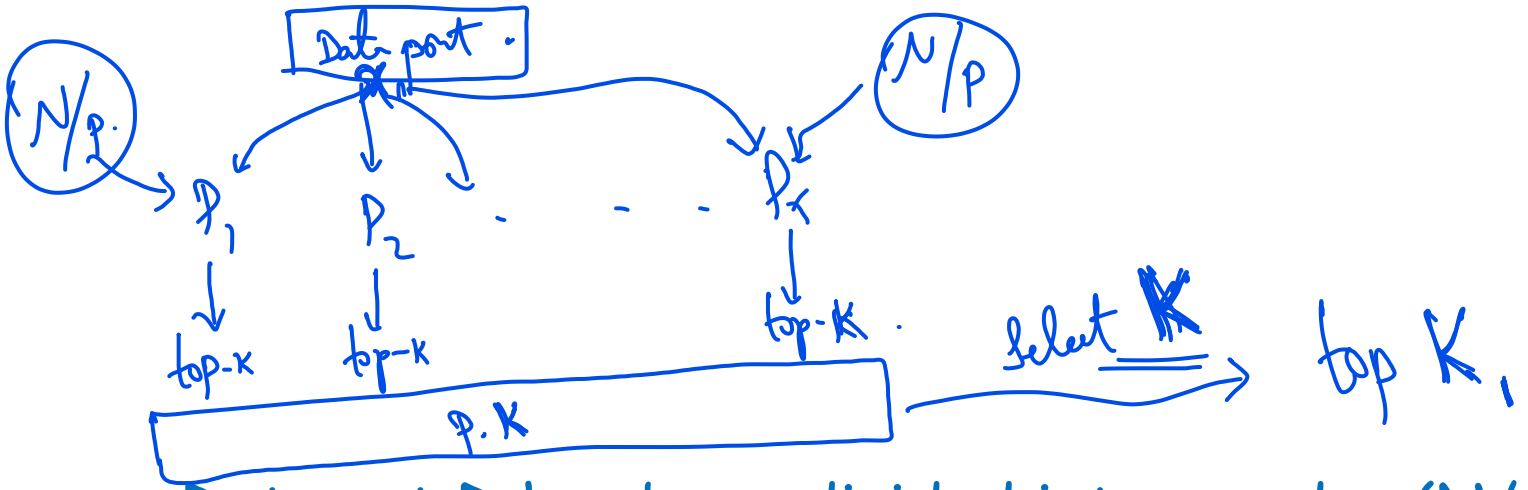
      Communication cost? Entire dataset is broadcast?
      Distributed KD-Tree Construction?

      N points into p nodes; (N/p) points each

Distributed algorithm (attempt 1)

- Distribute all points to p processsors(N/p points in each processors)
  - N values have to be communicated.
  - Communication cost: O(N)
- Step 1: Distance Computation
  - Transmit N/p values to p-1 processors.
  - Compute distances: O(N/p * N/p) computations.
- Step 2: Getting the nearest neighbors
  - Merge sort variant…

Cost:
N/p*N/p*p
Communication:
N^2 distances



- Data set D has been divided into p nodes (N/p points each; N = |D|)

- Let point x be in node i
- Send point x to all p nodes

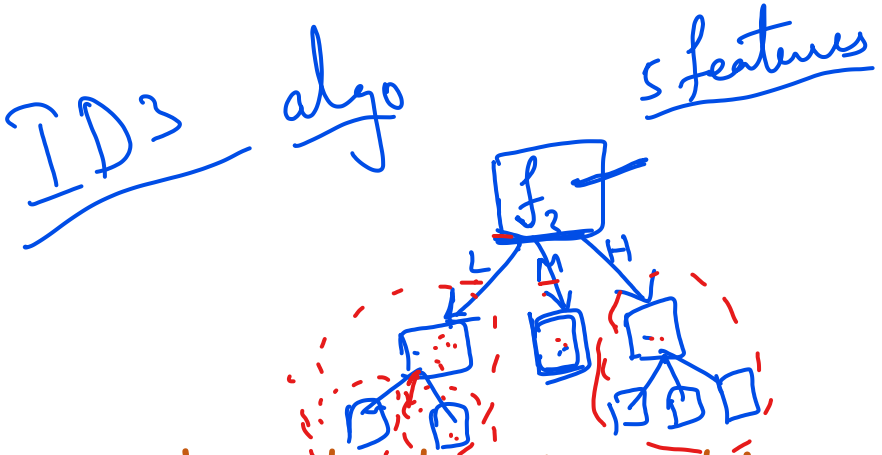
  - Parallel: In each node dist(x,y) is computed for all y in D_j j = 1 to p
  - Send back k nearest neighbors of x in node j to node I
  - node i has p*k potential neighbors; extract the k nearest

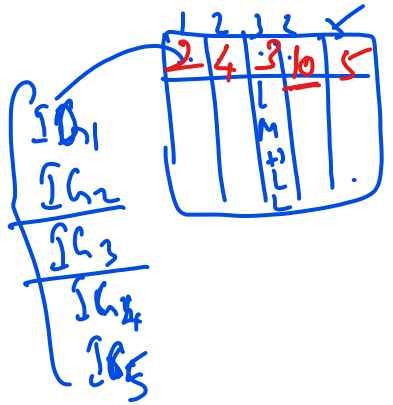
- O(k) per point

Communication Cost:
O(kN) distances

Decision Trees

ID3 algo





- Approach:
  - Construct a tree where each node denotes a binary decision
    - Nodes in the tree correspond to features and the order of features is chosen based on the notion of **information gain** (IG)
    - Information gain is the entropy
      - entropy of the whole set
      - minus the entropy when a particular feature is chosen

$f_3 \rightarrow 3(L, M, H)$, $d = 2$

f_x → A → d. Task parallelism → load imbalance.

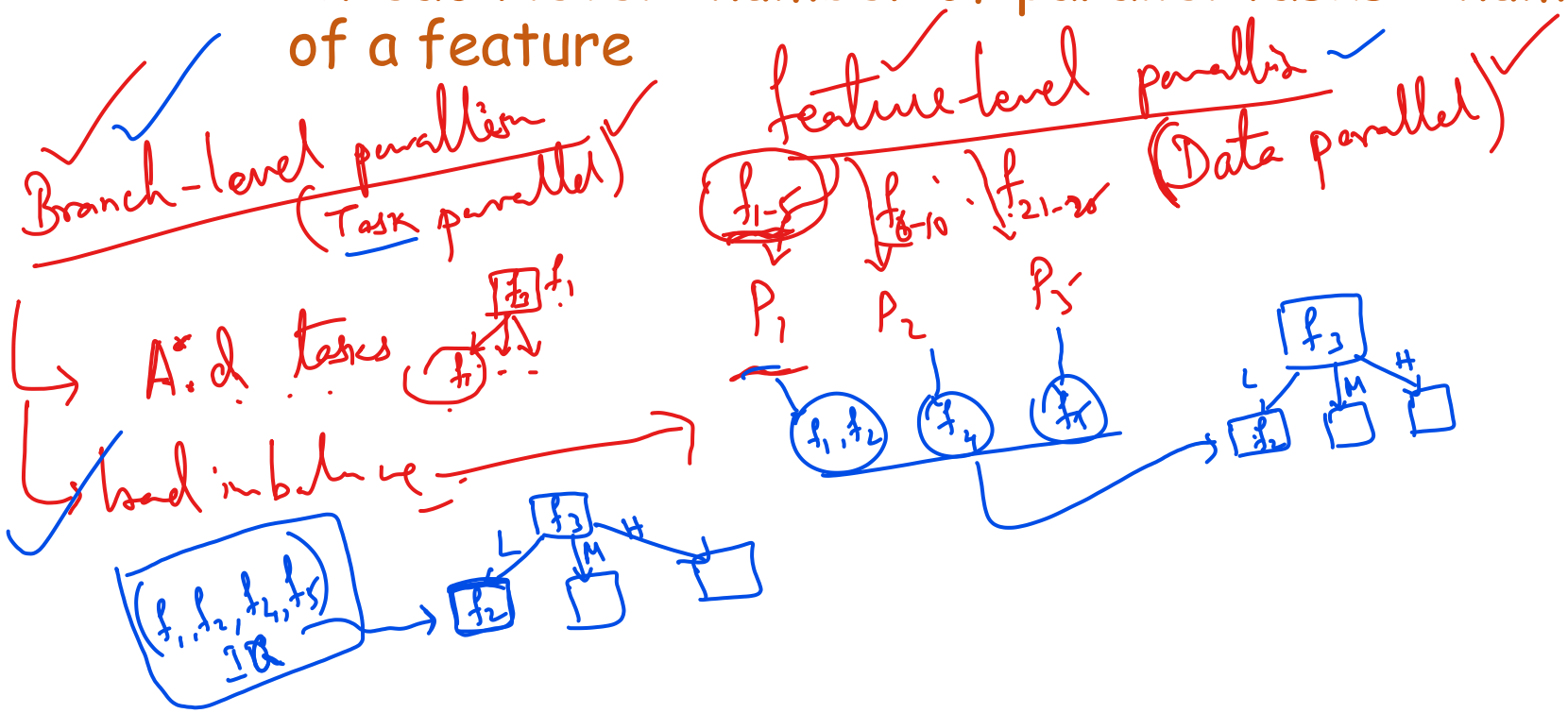
10 ---> 3 ⇒ 30 parallel tasks

Decision Tree Construction

- Algorithm ID3 (input dataset S)
  - If all examples have the same label
    - Return a leaf with that label
  - Else if there are no features left to test
    - Return a leaf with the most common label
  - Else choose the feature F that maximizes IG of dataset S as the next node
    - Add a branch from the node for each possible value f in F
    - For each branch:
      - Calculate S_f by removing F from the set of features
      - ID3(S_f)

Parallel/Distributed Tree Construction?

- When you branch assign each branch (corresponding to one value of a feature)
  - To a different task (task parallelism)
  - At each level : number of parallel tasks = number of possible values of a feature



Assignment

- Assignment will be released soon.
  - Assignment and Project will form a single end-to-end sequence of take-home team exercise involving:
    - Literature Survey,
    - Problem Formulation,
    - Design,
    - Implementation,
    - Testing and Demo
      - particularly for Performance
- Teams can include at most five students
  - Problems must focus on paralelization/distribution of ML algorithms
  - Target architectures can be multi-core, GPUs, or clusters or a combination thereof.
  - Any Programming language/environment or development platform may be used.