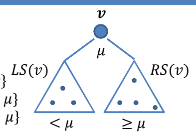
VP tree to Search for Nearest Neighbors within a Given Radius based on Triangle Inequality

Marzena Kryszkiewicz
Instytut Informatyki
Politechnika Warszawska

The Idea of Constructing a VP-Tree

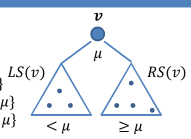
- **A node in VP-Tree contains**:
  - $v \in D$
  - $\mu = median(\{u \in S(v) | distance(u, v)\})$, where $S(v)$ is the subtree rooted in v}
  - $LS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) < \mu\}$
  - $RS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) \geq \mu\}$



- **Each point in *D* is stored only once in VP-Tree.**
- **Idea of how to select a point from *D* to be stored in the root of the VP-tree:**
- A point in the root of the VP-tree, say point *v*, should be the one with the maximal variance of its distances to all points in *D*. $LS(v)$ will be stored as the left subtree and $RS(v)$ - as the right subtree of the VP-tree.
- **Idea of how to select a point to be the root of a subtree covering a subset *D'* of points in *D*:**
- A point in the root of this subtree, say point *v*, should be the one with the maximal variance of its distances to all points in *D'*.

Practical Construction of a VP-Tree

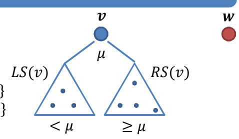
- **A node in VP-Tree contains**:
  - $v \in D$
  - $\mu = median(\{u \in S(v) | distance(u, v)\})$, where $S(v)$ is the subtree rooted in v}
  - $LS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) < \mu\}$
  - $RS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) \geq \mu\}$



- **Practical selection of a point from (a subset *D'* of) *D* to be stored in the root of a (sub-)tree:**
- A random sample of points from (subset *D'* of) *D* constitutes a set of candidates to be stored in the root of the (sub-)tree.
- Their medians and variances of distances are calculated with respect to another random sample of points from (subset *D'* of) *D*.
- The candidate point with the maximal variance of its distances to the points in the latter sample is stored in the root of the (sub-tree).
- The real median of this point is calculated based on its distances to all points in (subset *D'* of) *D,* and is also stored in the root of the (sub-)tree.

k/k⁺-NN Search in VP-Tree…

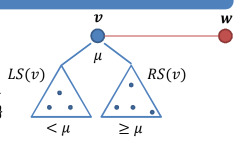
- **A node in VP-Tree contains**:
  - $v \in D$,
  - $\mu = median(\{u \in S(v) | distance(u, v)\})$
  - $LS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) < \mu\}$
  - $RS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) \geq \mu\}$



- **Search for k/k⁺-NN of point u within ε radius in node v of VP-Tree**:
  - $distance(w, v)$,
  - Cond. 1: $distance(w, v) - \mu \geq \varepsilon$. If true, then for each point u in $LS(v)$, $distance(w, v) - distance(u, v) > distance(w, v) - \mu \geq \varepsilon$. Thus, $distance(w, v) - distance(u, v) > \varepsilon$, so $LS(v)$ does not contain k/k⁺-NN(w) within the ε radius.
  - Cond. 2: $\mu - distance(w, v) > \varepsilon$. If true, then for each point u in $LS(v)$, $distance(u, v) - distance(w, v) \geq \mu - distance(w, v) > \varepsilon$. Thus, $distance(u, v) - distance(w, v) > \varepsilon$, so $LS(v)$ does not contain k/k⁺-NN(w) within ε radius.

k/k⁺-NN Search in VP-Tree

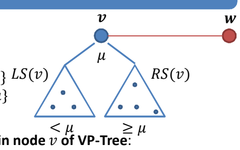
- **A node in VP-Tree contains**:
  - $v \in D$,
  - $\mu = median(\{u \in S(v) | distance(u, v)\})$
  - $LS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) < \mu\}$
  - $RS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) \geq \mu\}$



- **Search for k/k⁺-NN of point u within ε radius in node v of VP-Tree**:
  - $distance(w, v)$,
  - Cond. 1: $distance(w, v) - \mu \geq \varepsilon$. If true, then for each point u in $LS(v)$, $distance(w, v) - distance(u, v) > distance(w, v) - \mu \geq \varepsilon$. Thus, $distance(w, v) - distance(u, v) > \varepsilon$, so $LS(v)$ does not contain k/k⁺-NN(w) within the ε radius.
  - Cond. 2: $\mu - distance(w, v) > \varepsilon$. If true, then for each point u in $LS(v)$, $distance(u, v) - distance(w, v) \geq \mu - distance(w, v) > \varepsilon$. Thus, $distance(u, v) - distance(w, v) > \varepsilon$, so $LS(v)$ does not contain k/k⁺–NN(w) within ε radius.

Improved k/k⁺-NN Search in VP-Tree…

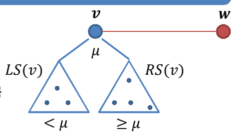
- **A node in VP-Tree contains**:
  - $v \in D$,
  - $\mu = median(\{u \in S(v) | distance(u, v)\})$
  - $LS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) < \mu\}$
  - $RS(v) = \{u \in S(v) \backslash \{v\} | distance(u, v) \geq \mu\}$



- **Search for k/k⁺-NN of point u within ε radius in node v of VP-Tree**:
  - $distance(w, v)$,
  - Cond. 1: $distance(w, v) - \mu \geq \varepsilon$. If true, then for each point u in $LS(v)$, $distance(w, v) - distance(u, v) > \varepsilon$, so kNN($w$) is not in $LS(v)$ within ε radius.
  - Cond. 2: $\mu - distance(w, v) > \varepsilon$. If true, then for each point u in $RS(v)$, $distance(u, v) - distance(w, v) > \varepsilon$, so kNN($w$) is not in $RS(v)$ within ε radius.
- **Improved search for k/k⁺-NN of point u within ε radius in node v of VP-Tree**:
  - Cond. 1': $distance(w, v) - left\_bound > \varepsilon$, where $left\_bound$ is the maximum of the distances from point v to all points in $LS(v)$.
  - Cond. 2': $right\_bound - distance(w, v) > \varepsilon$, where $right\_bound$ is minimum of the distances from point v to all points in $RS(v)$.

Improved k/k⁺-NN Search in VP-Tree

- **A node in VP-Tree contains**:
  - $v \in D$,
  - $\mu = median(\{u \in S(v) | distance(u, v)\})$
  - $LS(v) = \{u \in S(v) \setminus \{v\} | distance(u, v) < \mu\}$
  - $RS(v) = \{u \in S(v) \setminus \{v\} | distance(u, v) \geq \mu\}$



- **Improved search for k/k⁺-NN of point u within ε radius in node v of VP-Tree**:
  - Cond. 1’: $distance(w, v) - left\_bound > \varepsilon$, where $left\_bound$ is the maximum of the distances from point v to all points in $LS(v)$.
  - Cond. 2’: $right\_bound - distance(w, v) > \varepsilon$, where $right\_bound$ is minimum of the distances from point v to all points in $RS(v)$.

- **Example.** Let ε = 1, *left_bound*(*v*) = 8.5, *right_bound*(*v*) = 12 and *distance*(*w,v*) = 10. Then, *distance*(*w,v*) - *left_bound*(*v*) > ε and *right_bound*(*v*) - *distance*(*w,v*) > ε, which means that neither *LS*(*v*) nor *RS*(*v*) contains any nearest neighbor of *w* within the ε radius.

References

- Kryszkiewicz M., Janczak B.: Basic Triangle Inequality Approach Versus Metric VP-Tree and Projection in Determining Euclidean and Cosine Neighbors. Intelligent Tools for Building a Scientific Information Platform 2014: 27-49
- Moore, A. W.: The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data. In: Proc. of UAI, Stanford (2000) 397–405
- Yanilos P. N.: Data Structures and Algorithms of Nearest Neighbor Search in General Metric Spaces. Materiały z 4th ACM-SIAM Symposium on Descrete Algorithms, 1993, 311-321
- Zezula, P., Amato, G., Dohnal, V., Bratko, M.: Similarity Search: The Metric Space Approach. Springer (2006)