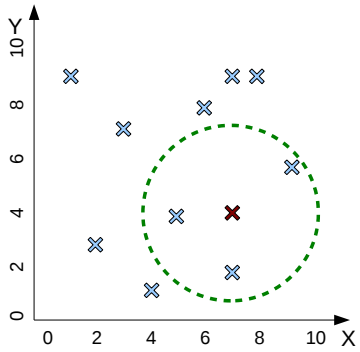
Why kNN is slow

What you see



What algorithm sees

- Training set:

  {(1,9), (2,3), (4,1),
  (3,7), (5,4), (6,8),
  (7,2), (8,8), (7,9), (9,6)}

- Testing instance:

  (7,4)

- Nearest neighbors?

  compare one-by-one
  to each training instance

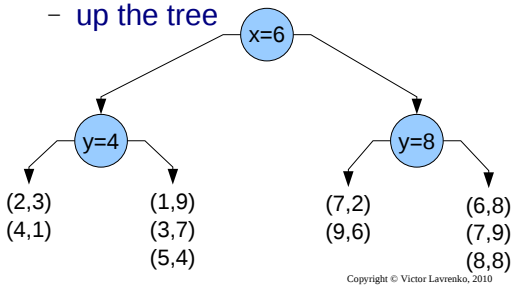
- n comparisons
- each takes d operations

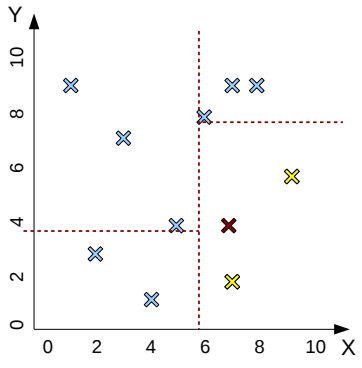
Making kNN fast

- Training: O(*1*), but testing: O (*nd*)
- Try to reduce *d*: reduce dimensionality
  - simple feature selection, since PCA/LDA etc. are O(d^3)
- Try to reduce *n*: don't compare to **all** training examples
  - idea: quickly identify *m* << *n* potential near neighbours
    - compare only to those, pick *k* nearest neighbours → O(*md*) time
  - **K-D trees**: low dimensionality, numeric data:
    - O(*d* log *n*), only works when *d* << *n,* inexact: may miss neighbours
  - **inverted lists**: high dimensionality, sparse data
    - O(*d' n'*), *d'*<<*d*, *n'*<<*n*, works for sparse (discrete) data, exact
  - **locality-sensitive hashing**: high-d, sparse or dense
    - O(*d H*), *H*<<*n*... number of hashes, inexact: may miss neighbours

K-D tree example

- Build a K-D tree:
  - {(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)}
  - pick random dimension, find median, split data points, repeat
- Find nearest neighbours for a new point: (7,4)
  - find region containing new point
  - compare to all points in region
  - up the tree



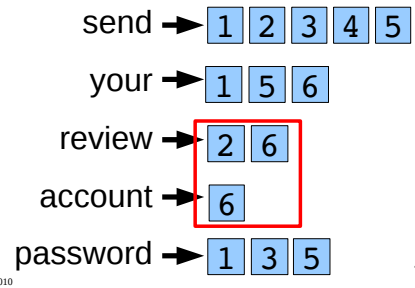


Inverted list example

- Data structure used by search engines (Google, etc.)
  - list of training instances that contain a particular attribute
  - main assumption: most attribute values are zero (sparseness)
- Given a new testing example:
  - fetch and merge inverted lists for attributes present in example
  - O(*n'd'*): *d'* … attributes present, *n'* … avg. length of inverted list

D1: “send your password”	spam
D2: “send us review”	ham
D3: “send us password”	spam
D4: “send us details”	ham
D5: “send your password”	spam
D6: “review your account”	ham

new email: “account review”



k Nearest Neighbours Summary

- Key idea: distance between training and testing point
  - important to select good distance function
- Can be used for classification and regression
- Simple, non-linear, asymptotically optimal
  - assumes only smoothness
  - “let data speak for itself”
- Value k selected by optimizing generalization error
- Naive implementation slow for large datasets
  - use K-D Trees (low d) or inverted indices (high d)