Case Study 2: Document Retrieval

Task Description: Finding Similar Documents

Announcements:

- HW1 due
- Project Milestones
  - Start early
  - Lit. review (>= 3 papers read carefully)
  - First rounds of experiments

- Today:
  - Review: Hash kernels
  - Today: similarity search, k-NNs, KD-trees
Document Retrieval

- **Goal:** Retrieve documents of interest
- **Challenges:**
  - Tons of articles out there
  - How should we measure similarity?

---

Task 1: Find Similar Documents

- **To begin...**
  - **Input:** Query article
  - **Output:** Set of $k$ similar articles
Document Representation

- Bag of words model

\[ X = \begin{bmatrix} w_1 & \cdots & w_d \end{bmatrix} \in \mathbb{R}^{n \times d} \]

“Bag of words” - word count of words; ignore word order.

©Sham Kakade 2017

Image Search...

Organic Authority
5 Bitter Melon Recipes: The Ancient Healing Fruit
bitter melon stir fry
Images may be subject to copyright.

Visit page Share
1-Nearest Neighbor

- Articles: $X = \{x^1, \ldots, x^N\}, \quad x^i \in \mathbb{R}^d$
- Query: $x \in \mathbb{R}^d$
- 1-NN
  - Goal: Find $x' \in K \setminus \{x\}$ closest to $x$.
  - Formulation:
    $$d(x', x) = \min_{x' \in K \setminus \{x\}} d(x', x)$$

$k$-Nearest Neighbor

- Articles: $X = \{x^1, \ldots, x^N\}, \quad x^i \in \mathbb{R}^d$
- Query: $x \in \mathbb{R}^d$
- $k$-NN
  - Goal: Find $k$ closest elements in $X$.
  - Formulation: 
Distance Metrics – Euclidean

\[ d(u, v) = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2} \]

Or, more generally,

\[ d(u, v) = \sqrt{\sum_{i=1}^{d} \sigma_i^2 (u_i - v_i)^2} \]

Equivalently,

\[ d(u, v) = \sqrt{(u - v)' \Sigma (u - v)} \]

where \( \Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_d^2 \end{bmatrix} \)

Other Metrics...
- Mahalanobis, Rank-based, Correlation-based, cosine similarity...

Notable Distance Metrics
(and their level sets)

\[ L_1 \] norm (absolute)

\[ L_\infty \] norm (max)

Scaled Euclidian (\( L_2 \))

Mahalanobis
(\( \Sigma \) is general sym pos def matrix, on previous slide = diagonal)
Euclidean Distance + Document Retrieval

- Recall distance metric
  \[ d(u, v) = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2} \]

- What if each document were \( \alpha \) times longer?
  - Scale word count vectors
  - What happens to measure of similarity?

- Good to normalize vectors

Issues with Document Representation

- Words counts are bad for standard similarity metrics

- Term Frequency – Inverse Document Frequency (tf-idf)
  - Increase importance of rare words

©Sham Kakade 2017
TF-IDF

- Term frequency:
  \[ tf(t, d) = \frac{\text{# occurrences of } t \text{ in } d}{\text{term length}} \]
  - Could also use \( \{0, 1\} \cdot 1 + \log f(t, d) \).
- Inverse document frequency:
  \[ \text{idf}(t, D) = \frac{N}{\log |\{d \in D : t \in d\}|} \]
- tf-idf:
  \[ \text{tfidf}(t, d, D) = tf(t, d) \times idf(t, D) \]
  - High for document \( d \) with high frequency of term \( t \) (high "term frequency") and few documents containing term \( t \) in the corpus (high "inverse doc frequency")

Issues with Search Techniques

- Naïve approach:
  - Brute force search
    - Given a query point \( x \)
    - Scan through each point \( x^i \)
    - \( O(N) \) distance computations per 1-NN query!
    - \( O(N\log k) \) per \( k \)-NN query!

- What if \( N \) is huge???
  (and many queries)
Think about Web Search/Image Search

- How big is $N$?

- How fast do we desire to do recall?

Intuition (?): NN in 1D and Sorting

- How do we do 1-NN searches in 1 dim?

- Pre-processing time:
  - $O(N)$
  - $O(N \log N)$

- Query time:
  - $O(1)$
  - $O(\log N)$
Smarter approach: **kd-trees**

- Structured organization of documents
  - Recursively partitions points into axis aligned boxes.
- Enables more efficient pruning of search space
  - Examine nearby points first.
  - Ignore any points that are further than the nearest point found so far.

**kd-trees** work “well” in “low-medium” dimensions

- We’ll get back to this...

---

**KD-Tree Construction**

- Start with a list of $d$-dimensional points.
KD-Tree Construction

- Split the points into 2 groups by:
  - Choosing dimension $d_j$ and value $V$ (methods to be discussed...)
  - Separating the points into $x_{d_j}^i > V$ and $x_{d_j}^i <= V$.

- Consider each group separately and possibly split again (along same/different dimension).
  - Stopping criterion to be discussed...
Consider each group separately and possibly split again (along same/different dimension).

- Stopping criterion to be discussed...

Continue splitting points in each set

- creates a binary tree structure
- Each leaf node contains a list of points
Keep one additional piece of information at each node:

- The (tight) bounds of the points at or below this node.

Use heuristics to make splitting decisions:
- Which dimension do we split along?
- Which value do we split at?
- When do we stop?
Many heuristics...

median heuristic

center-of-range heuristic

Nearest Neighbor with KD Trees

- Traverse the tree looking for the nearest neighbor of the query point.
Examine nearby points first:
- Explore branch of tree closest to the query point first.
When we reach a leaf node:
- Compute the distance to each point in the node.
Then backtrack and try the other branch at each node visited

Each time a new closest node is found, update the distance bound
Using the distance bound and bounding box of each node:
- Prune parts of the tree that could NOT include the nearest neighbor
Nearest Neighbor with KD Trees

- Using the distance bound and bounding box of each node:
  - Prune parts of the tree that could NOT include the nearest neighbor

Complexity

- For (nearly) balanced, binary trees...
- Construction
  - Size:
  - Depth:
  - Median + send points left right:
  - Construction time:
- 1-NN query
  - Traverse down tree to starting point:
  - Maximum backtrack and traverse:
  - Complexity range:

- Under some assumptions on distribution of points, we get $O(\log N)$ but exponential in $d$ (see citations in reading)
Complexity

Complexity for $N$ Queries

- Ask for nearest neighbor to each document
- Brute force 1-NN:
- $kd$-trees:
Inspections vs. $N$ and $d$

K-NN with KD Trees

- Exactly the same algorithm, but maintain distance as distance to furthest of current $k$ nearest neighbors
- Complexity is:
Approximate K-NN with KD Trees

- **Before:** Prune when distance to bounding box >
- **Now:** Prune when distance to bounding box >
- Will prune more than allowed, but can guarantee that if we return a neighbor at distance $r$, then there is no neighbor closer than $r / \alpha$.
- In practice this bound is loose...Can be closer to optimal.
- Saves lots of search time at little cost in quality of nearest neighbor.

Cover trees (+ ball trees)

- What about exact NNs searches in high dimensions?
- Idea: utilize triangle inequality of metric (so allow for arbitrary metric)
- cover-tree guarantees:
Cover trees: what does the triangle inequality imply?

Cover trees: data structure
Wrapping Up – Important Points

kd-trees
- Tons of variants
  - On construction of trees (heuristics for splitting, stopping, representing branches...)
  - Other representational data structures for fast NN search (e.g., cover trees, ball trees,...)

Nearest Neighbor Search
- Distance metric and data representation are crucial to answer returned

For both...
- High dimensional spaces are hard!
  - Number of kd-tree searches can be exponential in dimension
    - Rule of thumb... $N \gg 2^d$... Typically useless.
  - Distances are sensitive to irrelevant features
    - Most dimensions are just noise → Everything equidistant (i.e., everything is far away)
    - Need technique to learn what features are important for your task

What you need to know

- Document retrieval task
  - Document representation (bag of words)
  - tf-idf

- Nearest neighbor search
  - Formulation
  - Different distance metrics and sensitivity to choice
  - Challenges with large $N$

- kd-trees for nearest neighbor search
  - Construction of tree
  - NN search algorithm using tree
  - Complexity of construction and query
  - Challenges with large $d$
Acknowledgment

- This lecture contains some material from Andrew Moore’s excellent collection of ML tutorials:
  - [http://www.cs.cmu.edu/~awm/tutorials](http://www.cs.cmu.edu/~awm/tutorials)
- In particular, see:
  - [http://grist.caltech.edu/sc4devo/.../files/sc4devo_scalable_datamining.ppt](http://grist.caltech.edu/sc4devo/.../files/sc4devo_scalable_datamining.ppt)