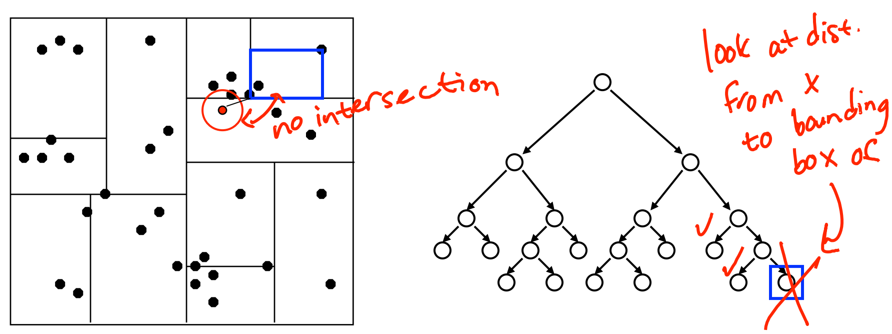
Case Study 2: Document Retrieval

Finding Similar Documents Using Nearest Neighbors

Machine Learning/Statistics for Big Data
CSE599C1/STAT592, University of Washington
Emily Fox

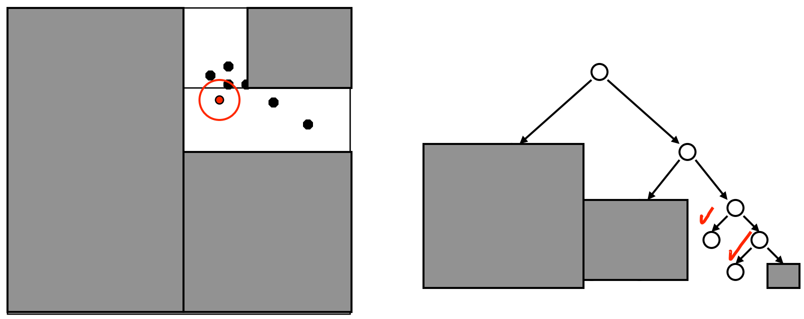
January 22^{nd}, 2013

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

Nearest Neighbor with KD Trees

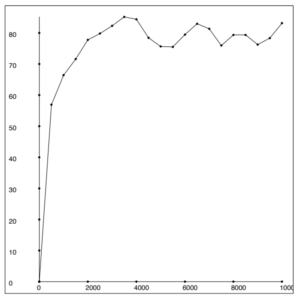


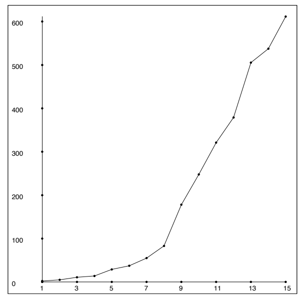
- Using the distance bound and bounding box of each node:
  - Prune parts of the tree that could NOT include the nearest neighbor

Complexity

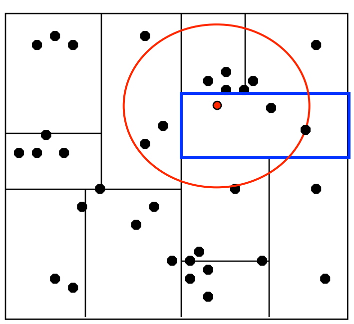
- For (nearly) balanced, binary trees...
- Construction
  - Size: $2N-1 \rightarrow O(N)$
  - Depth: $O(\log N)$
  - Median + send points left right: $O(N)$ at every tree level
  - Construction time: $O(N \log N)$ (smart)
- 1-NN query
  - Traverse down tree to starting point: $O(\log N)$
  - Maximum backtrack and traverse: $O(N)$ worst case
  - Complexity range: $O(\log N) \rightarrow O(N)$
- Under some assumptions on distribution of points, we get O(log*N*) but exponential in *d* (see citations in reading)

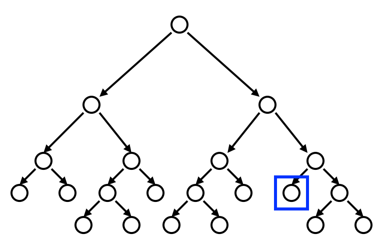
Inspections vs. *N* and *d*





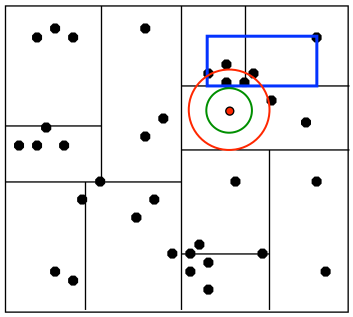
K-NN with KD Trees

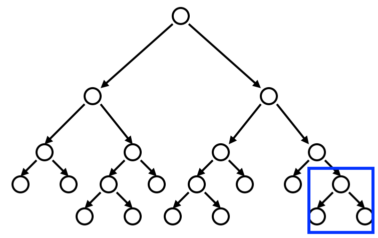




- Exactly the same algorithm, but maintain distance as distance to furthest of current *k* nearest neighbors
- Complexity is:

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/α.
- In practice this bound is loose…Can be closer to optimal.
- Saves lots of search time at little cost in quality of nearest neighbor.

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., 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… $N >> 2^d$… 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*

Locality-Sensitive Hashing
Hash Kernels
Multi-task Learning

Machine Learning/Statistics for Big Data
CSE599C1/STAT592, University of Washington

Carlos Guestrin

January 24th, 2013

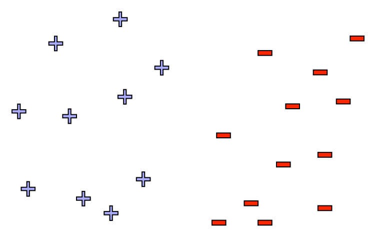
Using Hashing to Find Neighbors

- KD-trees are cool, but…
  - Non-trivial to implement efficiently
  - Problems with high-dimensional data
- Approximate neighbor finding…
  - Don't find exact neighbor, but that's OK for many apps, especially with Big Data
- What if we could use hash functions:
  - Hash elements into buckets:
  - Look for neighbors that fall in same bucket as **x**:
- But, by design…

Locality Sensitive Hashing (LSH)

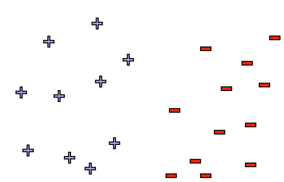
- A LSH function *h* satisfies (for example), for some some similarity function *d*, for r>0, α>1:
  - d(**x**,**x**') ≤ r, then P(h(**x**)=h(**x**')) is high
  - d(**x**,**x**') > α.r, then P(h(**x**)=h(**x**')) is low
  - (in between, not sure about probability)

Random Projection Illustration



- Pick a random vector **v**:
  - Independent Gaussian coordinates
- Preserves separability for most vectors
  - Gets better with more random vectors

Multiple Random Projections: Approximating Dot Products

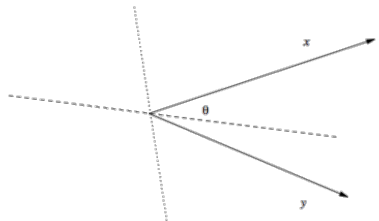


- Pick *m* random vectors $\mathbf{v}^{(i)}$:
  - Independent Gaussian coordinates
- Approximate dot products:
  - Cheaper, e.g., learn is smaller *m* dimensional space
- Only need logarithmic number of dimensions!
  - *N* data points, approximate dot-product within ε>0:

$$m = \mathcal{O}\left(\frac{\log N}{\epsilon^2}\right)$$

- But all sparsity is lost

LSH Example: Sparser Random Projection for Dot products



- Pick random vectors $\mathbf{v}^{(i)}$
- Simple 0/1 projection: $h_i(x) =$
- Now, each vector is approximated by a bit-vector
- Dot-product approximation:

LSH for Approximate Neighbor Finding

- Very similar elements fall in exactly same bin:
- And, nearby bins are also nearby:
- Simple neighbor finding with LSH:
  - For bins *b* of increasing hamming distance to h(**x**):
    - Look for neighbors of **x** in bin *b*
  - Stop when run out of time
- Pick *m* such that $N/2^m$ is "smallish"

Hash Kernels: Even Sparser LSH for Learning

- Two big problems with random projections:
  - Data is sparse, but random projection can be a lot less sparse
  - You have to sample *m* huge random projection vectors
    - And, we still have the problem with new dimensions, e.g., new words
- **Hash Kernels**: Very simple, but powerful idea: combine sketching for learning with random projections
- Pick 2 hash functions:
  - *h* : Just like in Min-Count hashing
  - ξ : Sign hash function
    - Removes the bias found in Min-Count hashing (see homework)
- Define a "kernel", a projection ϕ for **x**:

Hash Kernels, Random Projections and Sparsity

$$\phi_i(\mathbf{x}) = \sum_{j:h(j)=i} \xi(j)\mathbf{x}_j$$

- Hash Kernel as a random projection:
- Random projection vector for coordinate *i* of ϕ_i:
- Implicitly define projection by *h* and ξ, so no need to compute a priory and automatically deal with new dimensions
- Sparsity of ϕ, if **x** has *s* non-zero coordinates:

Hash Kernels Preserve Dot Products

- Hash kernels provide unbiased estimate of dot-products!
- Variance decreases as O(1/*m)*
- Choosing *m*? For ε>0, if

$$m = \mathcal{O}\left(\frac{\log \frac{N}{\delta}}{\epsilon^2}\right)$$

  - Under certain conditions…
  - Then, with probability at least 1-δ:

$$(1-\epsilon)||\mathbf{x}-\mathbf{x}'||_2^2 \leq ||\phi(\mathbf{x})-\phi(\mathbf{x}')||_2^2 \leq (1+\epsilon)||\mathbf{x}-\mathbf{x}'||_2^2$$

Learning With Hash Kernels

- Given hash kernel of dimension *m*, specified by *h* and ξ
  - Learn *m* dimensional weight vector
- Observe data point **x**
  - Dimension does not need to be specified a priori!
- Compute ϕ(**x**):
  - Initialize ϕ(**x**)
  - For non-zero entries *j* of $\mathbf{x}_j$:
- Use normal update as if observation were ϕ(**x**), e.g., for LR using SGD:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{-\lambda w_i^{(t)} + \phi_i(\mathbf{x}^{(t)})[y^{(t)} - P(Y=1|\phi(\mathbf{x}^{(t)}), \mathbf{w}^{(t)})]\right\}$$

Interesting Application of Hash Kernels: Multi-Task Learning

- Personalized click estimation for many users:
  - One global click prediction vector **w**:
    - But…
  - A click prediction vector $\mathbf{w}_u$ per user *u*:
    - But…
- Multi-task learning: Simultaneously solve multiple learning related problems:
  - Use information from one learning problem to inform the others
- In our simple example, learn both a global **w** and one $\mathbf{w}_u$ per user:
  - Prediction for user *u*:
  - If we know little about user *u*:
  - After a lot of data from user *u*:

Problems with Simple Multi-Task Learning

- Dealing with new user annoying, just like dealing with new words in vocabulary
- Dimensionality of joint parameter space is HUGE, e.g. personalized email spam classification from Weinberger et al.:
  - 3.2M emails
  - 40M unique tokens in vocabulary
  - 430K users
  - 16T parameters needed for personalized classification!

Hash Kernels for Multi-Task Learning

- Simple, pretty solution with hash kernels:
  - Very multi-task learning as (sparse) learning problem with (huge) joint data point **z** for point **x** and user *u*:
- Estimating click probability as desired:
- Address huge dimensionality, new words, and new users using hash kernels:
  - Desired effect achieved if *j* includes both
    - just word (for global **w**)
    - word,user (for personalized $\mathbf{w}_u$)

Simple Trick for Forming Projection $\phi(\mathbf{x},u)$

- Observe data point **x** for user *u*
  - Dimension does not need to be specified a priori and user can be unknown!
- Compute $\phi(\mathbf{x},u)$:
  - Initialize $\phi(\mathbf{x},u)$
  - For non-zero entries *j* of $\mathbf{x}_j$:
    - E.g., j='Obamacare'
    - Need two contributions to ϕ:
      - Global contribution
      - Personalized Contribution
    - Simply:
- Learn as usual using $\phi(\mathbf{x},u)$ instead of $\phi(\mathbf{x})$ in update function

Results from Weinberger et al. on Spam Classification: Effect of *m*

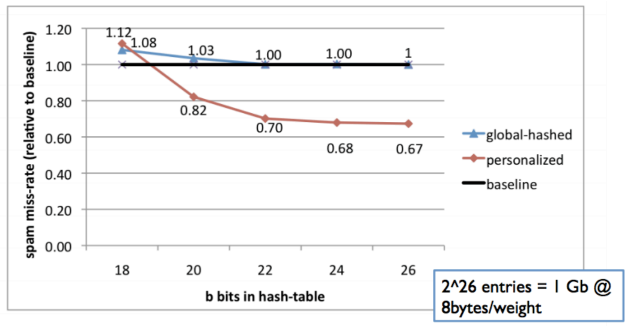


Figure 2. The decrease of uncaught spam over the baseline classifier averaged over all users. The classification threshold was chosen to keep the not-spam misclassification fixed at 1%. The hashed global classifier (*global-hashed*) converges relatively soon, showing that the distortion error ϵ_d vanishes. The personalized classifier results in an average improvement of up to 30%.

Results from Weinberger et al. on Spam Classification: Illustrating Multi-Task Effect

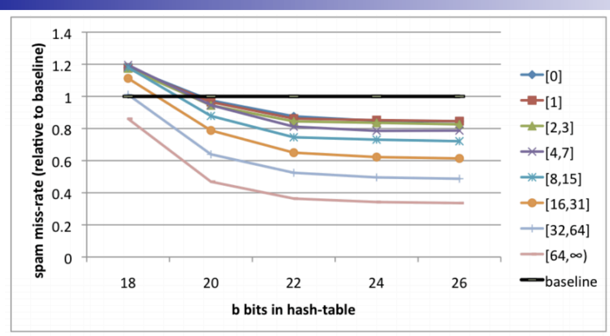


Figure 3. Results for users clustered by training emails. For example, the bucket $[8, 15]$ consists of all users with eight to fifteen training emails. Although users in buckets with large amounts of training data do benefit more from the personalized classifier (up-to 65% reduction in spam), even users that did not contribute to the training corpus at all obtain almost 20% spam-reduction.

What you need to know

- Locality-Sensitive Hashing (LSH): nearby points hash to the same or nearby bins
- LSH use random projections
  - Only O(log N/ε^2) vectors needed
  - But vectors and results are not sparse
- Use LSH for nearest neighbors by mapping elements into bins
  - Bin index is defined by bit vector from LSH
  - Find nearest neighbors by going through bins
- Hash kernels:
  - Sparse representation for feature vectors
  - Very simple, use two hash function
    - Can even use one hash function, and take least significant bit to define ξ
  - Quickly generate projection $\phi(\mathbf{x})$
  - Learn in projected space
- Multi-task learning:
  - Solve many related learning problems simultaneously
  - Very easy to implement with hash kernels
  - Significantly improve accuracy in some problems
    - if there is enough data from individual users

Case Study 2: Document Retrieval

Clustering Documents

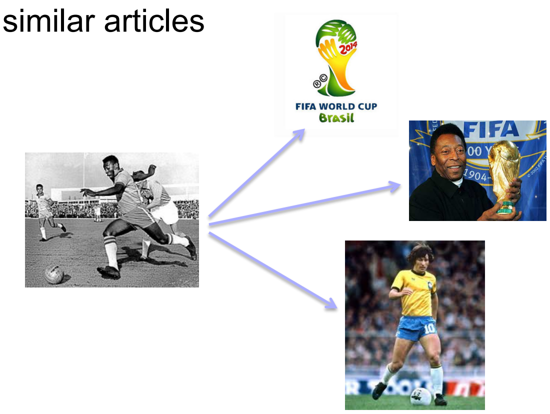
Machine Learning/Statistics for Big Data
CSE599C1/STAT592, University of Washington
Emily Fox
January 24th, 2013

Document Retrieval

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

Task 1: Find Similar Documents

- **So far…**
  - **Input:** Query article
  - **Output:** Set of k similar articles



Task 2: Cluster Documents

- **Now:**
  - Cluster documents based on topic

Document Representation

- Bag of words model

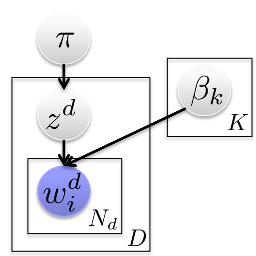
document *d*

A Generative Model

- Documents:
- Associated topics:
- Parameters: $\theta = \{\pi, \beta\}$

A Generative Model

- Documents: $x^1, \ldots, x^D$
- Associated topics: $z^1, \ldots, z^D$
- Parameters: $\theta = \{\pi, \beta\}$
- Generative model:



Form of Likelihood

- Conditioned on topic...

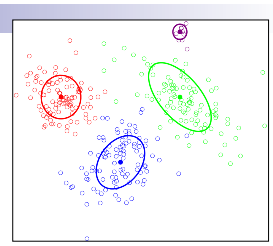
$$p(x^d \mid z^d, \beta) =$$

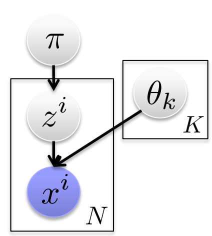
- Marginalizing latent topic assignment:

$$p(x^d \mid \beta, \pi) =$$

Gaussian Mixture Model

- Most commonly used mixture model
- Observations:
- Parameters:
- Likelihood:



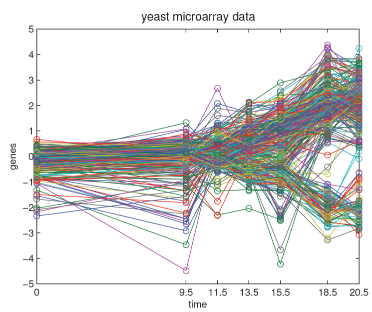


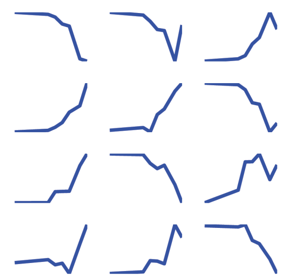
- Ex. z^i = country of origin, x^i = height of i[th] person
  - k^{th} mixture component = distribution of heights in country k

Another Example

(Taken from Kevin Murphy's ML textbook)

- Data: gene expression levels
- Goal: cluster genes with similar expression trajectories





Mixture models are useful for…

- Density estimation
  - Allows for multimodal density
- Clustering
  - Want membership information for each observation
    - e.g., topic of current document
  - Soft clustering:

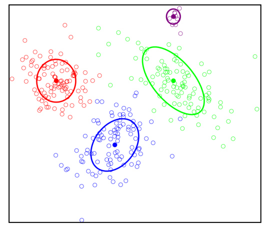
$$p(z^i = k \mid x^i, \theta) =$$

  - Hard clustering:

$$z^{i*} = \arg\max_k p(z^i = k \mid x^i, \theta) =$$

Issues

- Label switching
  - Color = label does not matter
  - Can switch labels and likelihood is unchanged



- Log likelihood is not convex in the parameters
  - No closed form gradient updates
  - Problem is simpler for “complete data likelihood”

- More on this next time…

What you need to know

- Mixture model formulation
  - Generative model
  - Likelihood