Announcements

Project #2 graded
- Individual grades on Canvas
- Team summaries through email
- Sparse comments are often a good thing

Course grades
- Projects 80%
- Participation (includes reading assignments) 20%

Project #3 is optional
- Can improve your project grade
- Can reform teams
- I can give names of other people looking for teammates...
- Teams should let me know by Friday if they are going to do Project #3

Announcements (II)

ASCSU surveys
- I need a volunteer
- I will end class 10 minutes early

Where were we?

Natural Language Processing
- Comparing document similarity
- Comparing words, ignoring syntax
- Every document is a point in term space
- Bag of Words approach

Latent Semantic Analysis (LSA)
- Models corpus as a Gaussian distribution of documents in term space
- Computes major axes of variance
- Compresses data
- Rule of thumb: keep 85% of variance
- Angle between vectors as similarity measure

Document matrices

Input: 2-d dimensional points
Output:
1st (right) singular vector: direction of maximal variance,
2nd (right) singular vector: direction of maximal variance, after removing the projection of the data along the first singular vector.
\( \sigma_1 \): measures how much of the data variance is explained by the first singular vector.
\( \sigma_2 \): measures how much of the data variance is explained by the second singular vector.
### Probabilistic LSA

Probabilistic Latent Semantic Analysis
- Essentially, a clustering technique
- Models data as a mixture of Gaussians
- Uses Expectation Maximization (EM) to...
- Fit cluster centers and \( \Sigma \) matrices (deviations)
- Assign cluster likelihoods to each sample
- Requires the number of clusters (\( K \)) as a parameter
- Project a sample into PLSA space:
  - Calculate probability of each cluster generating the sample
  - Normalizes to sum to 1
  - Sample exists
  - Converts probabilities to likelihoods
  - Resulting vector is point in PLSA space

### PLSA Example

Gaussian mixture
- Each Gaussian has a mean \( \mu \) and matrix of std. dev.'s \( \Sigma \)
- Probability of generating any sample can be computed from \( \mu \) and \( \Sigma \) using
  \[
  f(x) = \frac{1}{\sqrt{2\pi \det(\Sigma)}} e^{-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)}
  \]
- EM fits \( \mu_k \) and \( \Sigma_k \) for all \( k \) to maximize the probability of generating the training data

### Using PLSA

#### Training Data
- Run EM on training samples
- For every sample
  - For every cluster
    - Compute likelihood
  - Point becomes point in PLSA space
- Dimensions are cluster likelihoods
- Likelihood vector defines location

#### Test Data
- Compute likelihoods for trained clusters
- Becomes new point in PLSA space
- Compare to training data using Euclidean distance

### Nearest Neighbors

Goal: Find the nearest sample in a gallery to a novel probe sample

Obvious solution:
- Measure distance from probe to every gallery instance
- Select instance with smallest distance

Obvious problem:
- \( O(n) \)

### Approximate Nearest Neighbors

Goal: find nearest sample in gallery
- As often as possible
- When wrong, pick sample that is still close
- \( O(\log(n)) \)

Approach: binary trees
- Recursively divide feature space
- Each split divides gallery ~50/50

### ANN Illustrated


---

CS 510, IMAGE COMPUTATION, ©ROSS BEVERIDGE & BRUCE DRAPER
ANN Trees

Previous example thresholded feature values to divide feature space

Boundaries can be:
- Arbitrary hyperplanes (i.e. diagonal)
- Non-linear boundaries (i.e. spheres)
  - Example: Hierarchical K-Means

Problem:
- Samples near boundaries cause errors

Randomized Forests

Build multiple ANN trees
- With different boundaries
- Requires randomized boundary selection

Look up nearest neighbor in each tree
- Select best

Two versions in OpenCV
- Randomized Hierarchical K-Means
- FLANN

Proximity Forest

Problem: What if the feature space is unknown
- Imagine you have a similarity measure, but not a feature vector
- Examples
  - Similarity measure too expensive to run $O(n^2)$ times
  - Similarity measure over raw documents
  - Similarity measures over raw videos

Solution: proximity tree
- Select pivot sample at random
- Sort gallery by distance to pivot
- Split 50/50 nearest/farthest samples
- Repeat

Proximity Trees Illustrated

Two randomized proximity tree partitions

Proximity Forest Results

<table>
<thead>
<tr>
<th>Method</th>
<th>SIFT</th>
<th>KDT</th>
<th>HKM</th>
<th>MSER</th>
<th>PF</th>
</tr>
</thead>
</table>
| HKM    | 40%  | 50% | 60% | 70%  | 80%| 90%| 100%
| KDT SIFT|      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| HKM SIFT|      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| PF SIFT |      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| HKM MSER|      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| KDT MSER|      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| PF MSER |      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| HKM 3D  |      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| KDT 3D  |      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%
| PF 3D   |      |     |     |      | 40%| 50%| 60%| 70%| 80%| 90%| 100%

SIFT data: 128 dimensions, MSER data: 12 dimensions, 3D data: 3 dimensions

Source: O’Hara & Draper, Are You Using the Right Approximate Nearest Neighbor Algorithm?, WACV 2013

And the next step is...

Geometric Hashing!
- Create a function that maps samples to hash codes
- Similar samples should have similar codes
- Dissimilar samples should have different codes
- $O(1)$

- Not yet as accurate
  - Easy to map many training samples to same code
  - Devolves into linear search
  - Easy to map a new sample to a unique code
  - Useless
  - Necessary for really big data sets
  - E.g. Google image search